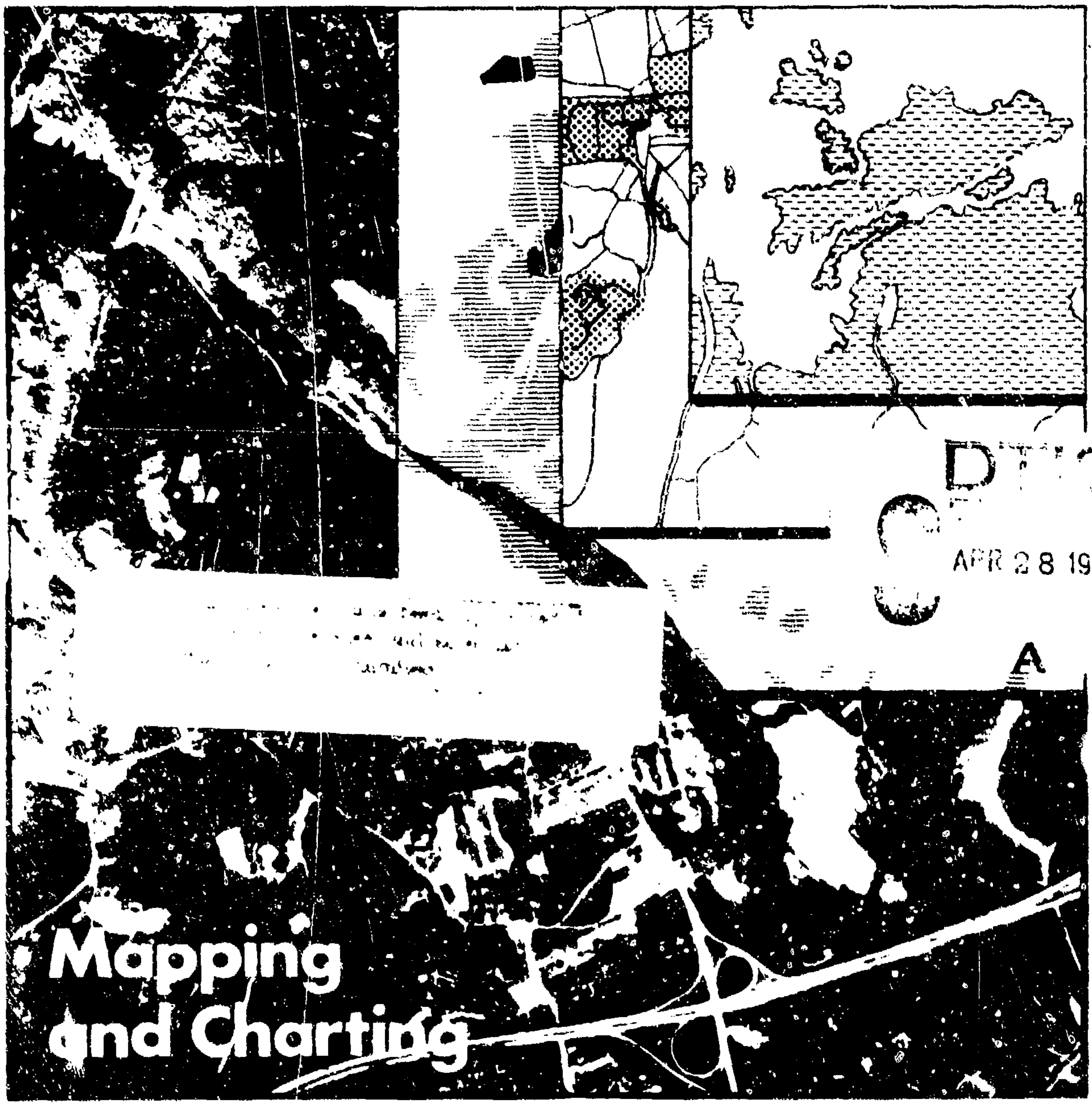
LEVEL (12)

IMAGE UNDERSTANDING WORKSHOP

APRIL 1981

Sponsored by:
Information Processing Techniques Office
Defense Advanced Research Projects Agency

Conducted in
Conjunction with:
Society of Photo-Optical
Instrumentation Engineers



Science Applications, Inc.

(6)

IMAGE UNDERSTANDING

Proceedings of a Workshop
Held at
Washington, D.C.
April 23, 1981.

(9) Semi-annual technical rept. Apr 81 - Apr 81

Sponsored by the
Defense Advanced Research Projects Agency

(11) Apr 81 (12) 256

(14)

Science Applications, Inc.
Report Number SAI-82-391-WA
Lee S. Baumann
(10) Workshop Organizer and
Proceedings Editor

(15) MDA903-81-C-0075
WDARPA Order-3456

This report was supported by
The Defence Advanced Research
Projects Agency under DARPA
Order No. 3456, Contract No. MDA903-81-C-0075
Monitored by the
Defense Supply Service, Washington, D.C.

408 401

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SAI-82-391-WA ✓	2. GOVT ACCESSION NO. AD-A098 261	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) IMAGE UNDERSTANDING Proceedings of a Workshop, April, 1981		5. TYPE OF REPORT & PERIOD COVERED SEMI ANNUAL TECHNICAL APRIL, 1980-APRIL, 1981
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) LEE S. BAUMANN (Ed.)		8. CONTRACT OR GRANT NUMBER(s) MDA903-81-C-0075
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS, INC. ✓ 1710 Goodridge Drive, 10th Floor McLean, Virginia 22102		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA ORDER No. 3456
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE APRIL, 1981
		13. NUMBER OF PAGES 245
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR RELEASE; DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digital Image Processing; Image Understanding; Scene Analysis; Edge Detection; Image Segmentation; CCDArrays; CCD Processors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This document contains the technical papers and outlines of semi-annual progress reports presented by the research activities in Image Understanding, sponsored by the Information Processing Techniques Office; Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 23 April 1981 in Washington, D.C.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

FOREWORD

The Twelfth Image Understanding Workshop sponsored by the Defense Advanced Research Projects Agency (DARPA), Information Processing Techniques Office was held in Washington, D.C. on April 23, 1981. The Workshop was conducted in conjunction with the Technical Symposium East'81 organized by the Society of Photo-Optical Instrumentation Engineers (SPIE).

Lt. Col. Larry E. Druffel, the DARPA program manager for Image Understanding, acted as the Workshop Chairman. In his opening remarks, Lt. Col. Druffel noted that he expected an exciting program for this years workshop. He advised the large audience that he was enthused by the level of technical progress made in the past year by the various research organizations in the DARPA sponsored program. Also, Druffel indicated that he was pleased with the association with SPIE as this provided an opportunity for interchange of ideas between cooperating groups which could only lead toward improvements in the ongoing research programs. The SPIE special sessions on Techniques and Applications of Image Understanding held on Tuesday and Wednesday April 21-22, commented Druffel, are indicative of the growing interest in Image Understanding and the maturing nature of I.U. science. We believe this to be an excellent opportunity, he noted, for researchers to percieve the potential uses for I.U.; and for a larger audience to become aware of the current state-of-the-art.

During the workshop, fifteen technical papers were presented by members of the university and industrial organizations involved in the DARPA sponsored I.U. program. These papers, and others of current interest for which time was not available for presentation at the workshop, are contained in Section I of these proceedings. In order to reach a wider audience of interested users and research personnel, many of these papers have been provided to SPIE for inclusion in their symposium proceedings as well as being printed in this DARPA workshop issue. Section II of this volume contains brief reviews prepared by the principal investigators involved in the DARPA sponsored research which, although not presented due to the press of time, are meant to keep the various members of the group, as well as those government personnel who have been interested in the I.U. program, abreast of the thrust of the research efforts being undertaken at each location.

Readers are reminded that extra copies of these proceedings as well as copies of previous proceedings may be secured from the Defense Technical Information Center, Cameron Station, Alexandria, Virginia 22314. Accession numbers for past editions are as shown on following page.

The materials for the cover of this document were provided by Mr. Bruce Opitz of the Advanced Technology Division, Headquarters Defense Mapping Agency. The accompanying description reads:

> Imagery is the primary source used by the Defense Mapping Agency (DMA). Many different kinds of information may be extracted depending on which products are required over each area. Currently, this photo-interpretation function is almost totally manual. As the extraction process becomes increasingly automated, DMA can begin to extract all possible needed information over an area, whether it is currently required or not, and stored in a universal data base. Any products tailored to each individual user's needs could then be synthesized from this data base.

Mr. Tom Dickerson, Science Applications, Inc. did the layout for the cover design. Appreciation is also due to Miss Ann Kastris of Science Applications, Inc. for assistance in putting together these proceedings and for handling the invitations and mailings necessary to the conduct of the workshop. Her valuable assistance in registration and the myriad of other details was also a sine qua non for this undertaking. Finally, our thanks to the Society of Photo-Optical Instrumentation Engineers for their cooperation and assistance during the planning and execution of this combined endeavor. We hope that they are in agreement with us that it has been a valuable experience.

Lee S. Baumann
Science Applications, Inc.
Workshop Organizer

DEFENSE TECHNICAL INFORMATION CENTER

ACCESSION NUMBERS FOR PREVIOUS

I.U. WORKSHOPS

ISSUE	DATE	NUMBER
APRIL	1977	ADA 052900
OCTOBER	1977	ADA 052901
MAY	1978	ADA 052902
NOVEMBER	1978	ADA 064765
APRIL	1979	ADA 069515
NOVEMBER	1979	ADA 077568
APRIL	1980	ADA 084764

AUTHOR INDEX

AUTHOR INDEX (Cont'd)

SECTION I

TECHNICAL PAPERS

REASONING ABOUT IMAGES: APPLICATION TO AERIAL IMAGE UNDERSTANDING

Peter G. Selfridge
Kenneth R. Sloan, Jr.

Department of Computer Science
University of Rochester
Rochester, New York 14627

Abstract

Image Understanding (I.U.) shares with Artificial Intelligence the need for mechanisms of using knowledge to control computation. In I.U., this knowledge takes the form of prior knowledge about objects and knowledge gained from doing the computations themselves. This paper presents an approach to the general I.U. problem and a specific program for locating buildings in aerial photographs. Prior knowledge is stored as an Appearance Model, which represents the appearances of possible buildings. A three stage program, starting with an Appearance Model Expert, generates operator and parameter sequences to achieve recognition. Some operators are Adaptive Operators, which hill-climb on a single parameter to optimize a feature match. Each level uses partial results from below to 1) search parameters for the best match, 2) infer obscuring image conditions and deal with them, and 3) infer high-level conditions such as the presence of occluding objects.

Introduction

Image Understanding (I.U.) shares many problems with the field of Artificial Intelligence (A.I.). The use of knowledge to control processing is one such general problem. In an I.U. domain, this can be formulated as the problem of using both prior knowledge about the task domain, and knowledge acquired as processing proceeds, to choose a sequence of computations which will achieve recognition and location of the desired objects.

We have formalized the I.U. problem in the following way. Prior object knowledge is stored in an Appearance Model, a graph structure encoding the expected appearance of the objects. Computational Primitives are routines that compute on images or derived data structures. The I.U. problem is to find a correspondence between sub-graphs of the appearance model and subsets (regions) of the input image, using the available computational primitives.

It is a tenet of this research that a program to do this must be prepared to use partial results, such as partial matches, to change the sequence of operators and their parameters. In doing so, it must be able to evaluate the results of its processing and be prepared to infer, at several levels, what image conditions are impeding recognition of the desired objects.

This paper describes a program to locate buildings in aerial photographs using the above approach. The program has three parts, shown in Figure 1, each of which embodies different levels of knowledge and inference ability.

An Appearance Model Expert chooses subgraphs of the building Appearance Model. A Region Description is derived from this sub-model and passed to the Operator and Image Problem Expert. This Expert chooses operators to compute candidate regions and calls other operators to try to match the region characteristics with the Region Description. One kind of operator is an Adaptive Operator, which can vary a single parameter to optimize the match with a specific value or range of a feature. The Operator and Image Problem Expert can infer the existance of a possible image problem and take appropriate action. The Appearance Model Expert can infer high-level conditions and check spatial relations between regions to recognize buildings complexes.

Appearance Models and The Appearance Model Expert

An Appearance Model is a data structure encoding the expected appearances of a class of objects. This model could be derived by a "smart" modeller from many sources of prior knowledge such as a light model, a camera model, and a 3D domain model. The "appearance primitive" in our model is a region. A simplified Appearance Model for buildings is shown in Figure 2, where sub-model nodes are represented as rectangles, property nodes as ovals, and property value nodes (place-holders for actual values) as double ovals.

The Appearance Model Expert uses rule-based knowledge of the semantics of the Appearance Model, and any partial results achieved so far, to choose sub-models representing the current goal. For example, some sample rules in English are:

```
If no Candidates then
    pick "Easiest" sub-model

If Current-Model decomposable then
    locate Sub-Components
    Check Interrelations

If Partial-Success then
    check "close" other models

If Occlusion-Indicated then
    locate possible Occluding object
```

These rules result in a subgraph being choosen. From that subgraph, a Region Description (RD) is created by taking all the region property values from that subgraph. This RD is a representation of the desired object in terms of expected region properties. This RD is passed to the Operator and Image Problem Expert.

Operator and Image Problem Expert

The Operator and Image Problem Expert is given a Region Description from the Appearance Model Expert, and attempts to locate a region with the desired properties. To do this, it invokes appropriate operators to generate candidate regions, and calls Adaptive Operators to try to achieve good matches on specific properties.

As will be described, this process can also infer possible image problems from partial matches. Once a problem is infered, an operator can be choosen to alleviate the condition, or more information can be computed and the Expert can try again with the new constraints.

Adaptive Operators

An Adaptive Operator is a program that is given a candidate region and a desired range of property values for a single property. This represents a goal to the operator of computing a derived region from the candidate with its property as close to the desired range as possible. Each Adaptive Operator has knowledge of the effects of varying a single parameter of a single operator. An Adaptive Operator uses this knowledge to vary the parameter, creating derived regions from the candidate, to get a region that is the best match possible.

For example, Adaptive-Match-On-Size knows that for a dark region created by thresholding, raising that threshold will shrink the size of the region, and lowering it will increase the size. It uses that knowledge to get the region size as close as possible to a given size range.

Partial Results and Image Problems

Partial results are dealt with at each level in our system. Adaptive Operators deal with partial results in the form of a non-optimal match by attemting to derive regions that are a better match to a given property.

Another kind of partial result dealt with by the Operator and Image Problem Expert is when a candidate region is missing some essential feature. For example, if after applying all relevant computations in attempting to find a rectangular region, the resulting region has only three good corners, a hypothesis will be made that a Merging condition is present in the image. Merging is one of a number of domain independent image conditions that are handled at this level. Once this hypothesis is made, several alternatives can be considered. Basically, what happens is that model completion is done, resulting in a model of the merged building. This more complete model represents more known constraints. Other operators can now be brought to bear in an attempt to instantiate the refined model. Figure 3 illustrates a hypothetical example.

Finally, the Operator and Image Problem Expert may fail to achieve a perfect match. The Appearance Model Expert now has several alternatives, depending on the severity of the failure. The Appearance Model Expert may decide that a region computed from below is an instantiation of a sub-model other than the one under consideration, and it will attempt to verify that. It may also make a high-level inference, such as that a returned region was not a perfect match because it is being occluded by another object. It can then examine the Appearance Model which may indicate what the occluding object might be, and it can try to locate that object to verify the occusion.

An Example

This section illustrates a single example of the execution of our system, applied to a window of an aerial photograph. The partial Appearance Model used is one hypothetically derived, perhaps from knowledge of this particular site.

Appearance Model:

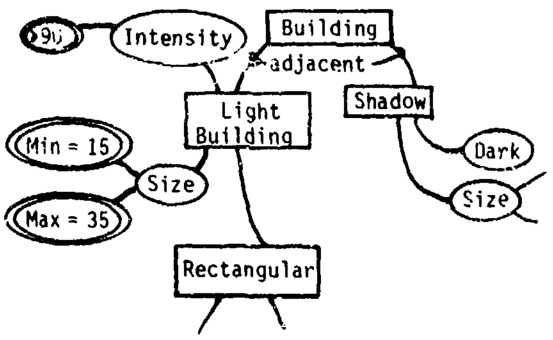


Image:

To instantiate this model, the Appearance Model Expert first attempts to locate the building by choosing the left sub-model and creating the following Region Description:

Region Description:
  Light
  Intensity > 90
  20 < Size < 40
  Rectangular: 4 corners tolerance 5.0
  4 sides tolerance 5.0

This RD is given to the Operator and Problem Expert, which calls operators to try to find a region matching this description.

Threshold (90) is called, creating this image:

The lower region is chosen as an initial candidate because its size is closest to that of the RD. Adaptive-Match-On-Size varies the threshold, creating the following series of images, with the threshold and size of the derived region below each figure:

1. Threshold = 172. Size = 11.

2. Threshold = 100 Size = 65.

3. Threshold = 150.
   Size = 14.

4. Threshold = 131
   Size = 19.

This candidate is now the right size. Adaptive-Rectangle attempts to see this region as a rectangle, and it succeeds. This result is passed to the Appearance Model Expert, who now tries to find the shadow, generating the following Region Description:

```
Region Description:
    Dark
    10 < Size < 20
    in the rectangle: 20, 5, 35, 20
```

The last attribute indicates to look in a rectangle around the located building. The Operator and Image Problem Expert calls Adaptive-Match-On-Size and again succeeds. The adjacency condition is checked, and succeeds as well. The two regions, corresponding to the located building and shadow, are outlined here:

Discussion

The example above illustrates two features of our system. First, it shows as Adaptive Operator at work generating derived regions to satisfy some criterion (region size). Second, it shows the Appearance Model Expert locating an object by parts, using a constraint (adjacency of parts) to limit search for the second part. It does not illustrate a current ability of the Operator and Image Problem expert: the detection of random noise and its application of an operator (local averaging) to alleviate it.

Other workers in this domain have written programs embodying some of the approachs of our program, and have achieved some good results[1, 2, 3, 4]. The analog of our Appearance Model is usually encoded as a fixed set of region properties, rather than a graph of different alternatives amenable to intelligent interpretation[1, 2]. Information from located objects is used in [1, 2] to refine a property table, which can result in a new segmentation. However, little use of partial matches is made, and none with the flexibility of our system.

The most novel features of our approach are the Adaptive Operators. While the hill-climbing techniques they embody are classic, indeed, amoung the oldest in A.I. [5], their application to an I.U. domain is new, as is their use in a framework of our kind. It is important to recognize that adaptive techniques, as all other computation, must be used in conjunction with an appropriate control structure. In our case, Adaptive Operators are applicable only if the candidate region is already "close".

The framework of our program provides a flexible rule-based system that applies operators when needed and relies on the Adaptive Operators to fine tune candidate regions. Each level in our program can evaluate and use the results from the level below, and deal with partial successes in a manner appropriate to that level.

Conclusion

Our program, as it stands, is still very sparse. It been run on six toy images like the one presented here. More interesting results await incorporation of more Computional Primitives and rules to guide them. Further details will be presented in [6].

References

1. Kestner, W., Bohner, M., Sharf, R., and M. Sties, "Object Guided Segmentation of Aerial Images", in Proceedings of the 5th International Conference on Pattern Recognition, pp. 529-531, 1980

2. Nago, M., Matsuyama, T., and H. Mori, "Structured Analysis of Complex Aerial Photographs", in Proceedings of the 6th International Joint Conference on Artificial Intelligence, pp. 610-616, 1977

3. Nago, M., Matsuyama, T., and Y. Ikeda, "Region Extraction and Shape Analysis of Aerial Photographs", in Proceedings of the 4th International Conference on Pattern Recognition, pp. 620-628, 1978

4. Nevatia, R. and K. Price, "Locating Structures in Aerial Images", in Proceedings of the 4th International Conference on Pattern Recognition, pp. 686-690, 1978

5. Selfridge, O., Howland, B., and M. Minsky, "Hill Climbing: Some Remarks on Multiple Optimization", Information and Control, 1959

6. P. Selfridge, "Reasoning About Images: Application to Aerial Image Understanding", forthcoming PhD thesis, 1981

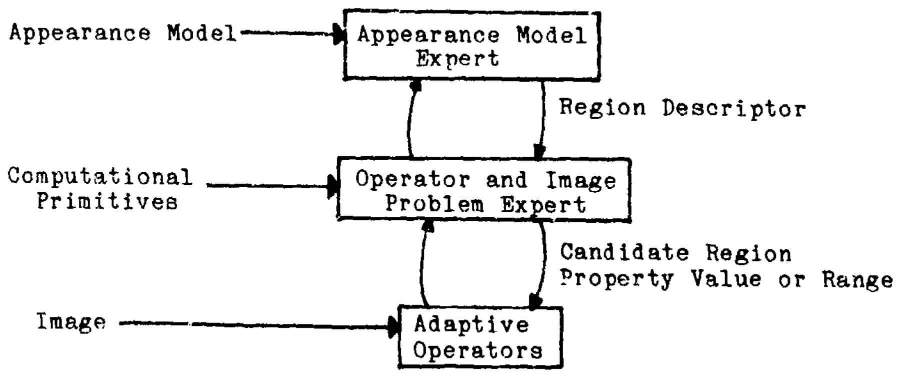


Figure 1: A Building Location Program

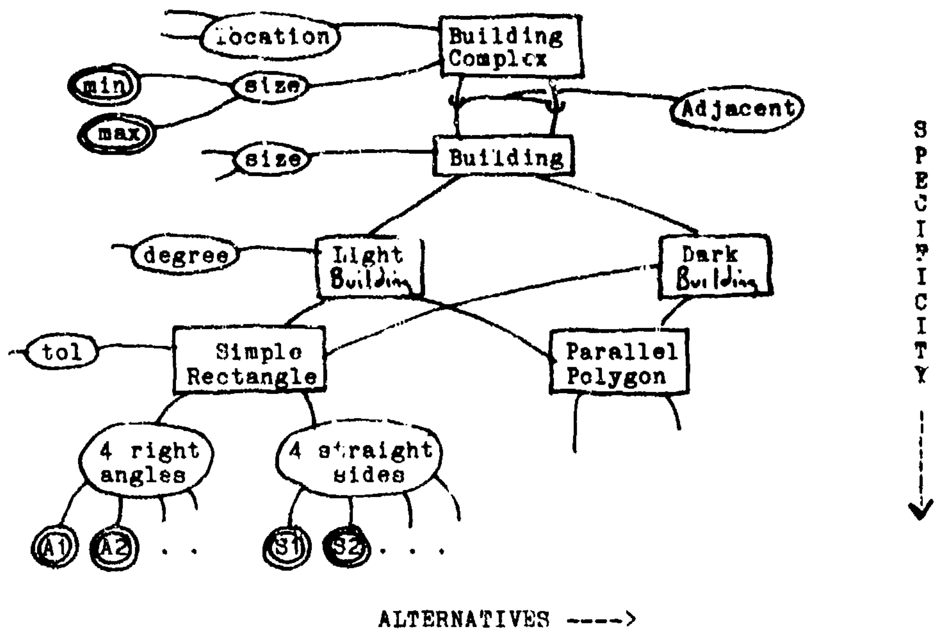
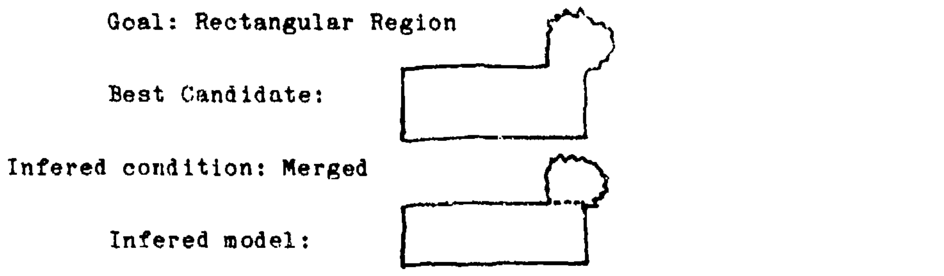


Figure 2: A Partial Appearance Model for Buildings in Aerial Photographs



New Constraint: Exact size and location of unknown corner

Possible actions: Run new operator, such as:
Edge-Finder
Adaptive-Match-On-Size
Seed-Base Region Grower

Figure 3: Model Completion by the Image Problem Expert

Database Support for Automated Photo Interpretation

David M. McKeown, Jr., and Takeo Kanade
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213*

1. Abstract

This paper is concerned with the use of a database to support automated photo interpretation. The function of the database is to provide an environment in which to perform photo interpretation utilizing software tools, and represent domain knowledge about the scenes being interpreted. Within the framework of the database, image interpretation systems use knowledge stored as map, terrain, or scene descriptions to provide structural or spatial constraints to guide human and machine processing. We describe one such system under development, MAPS (Map Assisted Photo interpretation System), and give some general rationales for its design and implementation.

2. Introduction

One characteristic of contemporary research in natural scene analysis is that knowledge about scene content is often highly integrated into the recognition program. The knowledge which has been most often implemented is *spectral* constraints; ie. characteristic *signatures* of water, vegetation, manmade objects etc., when detected by sensors having known signal response characteristics. Some examples of such sensors are, multi-channel infrared, radar (FLIR, SLR), and multi-spectral (mss) (LANDSAT, SEASAT, etc.). In many remote sensing applications, where the size (grain) of manmade physical objects is much smaller than the resolving power of the sensor, statistical techniques are employed to classify relatively large land areas. These techniques have found application in areas such as land use management, forestry, and geological mapping. However, pattern recognition and remote sensing techniques based purely on spectral analysis cannot handle those classes of imagery where individual feature detail is complex, and where shadows, occlusions and other 3D scene domain effects predominate.

In the work on **MAPS** discussed in this paper, we are interested in applying *spatial* and *structural* constraints to the interpretation of high resolution aerial mapping and satellite photographs. Briefly, these constraints can be used to determine *"where to look"* and *"what to look for"*. These constraints are represented as a *map database*. The map database itself is incrementally generated through interaction with the system, from human and machine segmentations of aerial imagery, and from collateral data. Images are registered to the existing map through an interactive correspondence procedure, in which a human operator specifies image-to-map correspondence guided by a landmark database. Once an initial correspondence has been obtained, it is possible to apply map domain knowledge to refine the correspondence and guide further image processing. It is this iterative procedure, using map knowledge to guide processing and assimilating results back into the map database, that is the core of future photo-interpretation systems. Further, as we begin to demonstrate the competency of automatic techniques for feature extraction, registration, and identification, these systems can gradually replace their interactive counterparts.

In the following sections we will examine several tasks in which spatial and structural constraints can be used to guide both *interactive* and *automated* photo interpretation. The major components of our system will be presented.

3. MAPS System Goals

MAPS is a collection of interactive display-oriented expert programs which represent and utilize map, terrain, and image data over a large area centered over Washington D.C. For a detailed discussion of the task domain and data representation and organization see [5].

There are several major goals of this research:

- Show that map knowledge can be incrementally compiled from a collection of time ordered aerial photographs. Such knowledge is composed of structural and spatial feature descriptions, and a large scale spatial organization (conceptual map).

*This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, and monitored by the Air Force Avionics Laboratory under Contract F33615-78-C-1551. The views and conclusions in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

- Demonstrate that map knowledge can be used to aid in the automatic interpretation of aerial imagery by providing *spatial* and *structural* constraints.

- Experiment with facilities which support data modeled at multiple levels of resolution, and which provide computation in the symbolic domain: conceptual map, terrain descriptions, and scene descriptions.

- Provide an integrated database access and display capability to allow users to view map, terrain and feature descriptions superimposed on image data.

3.1. System Data

The organization of a database system, which includes large numbers of complex imagery and collateral data, is a major research and design problem in its own right. The reader is referred to [5] for a discussion of primative data types and operations in mapping signal domain data into symbolic representations. The following section tabulates the signal data represented in MAPS.

- Image Data

  - Approximately 40 aerial mapping photographs providing temporal and spatial overlap. Scale ranges from 1:12000 to 1:36000, digitized at 100microns aperture to image format of 2048x2048 with 8 bits of intensity information per pixel.

  - Color and multi-spectral satellite imagery (6) in resolutions of 1:60000 and 1:1000000 for large area coverage at lower resolution.

  - Digitized images from United States Geologic Survey (USGS) topographic maps and common tourist guide maps. These images are currently used as a graphical interface to provide coverage information and cueing for users.

- Terrain Data

  - USGS digital terrain database, elevations over 3 second square grid in meters above sea level. Data is organized into 15 minute quadrants, each containing 90,000 points. Access to any grid point within the area bounded by <North 38deg, West 76deg> and <North 30deg, West 78deg>.

- Map Data

  - Defense Mapping Agency (DMA) radar simulation database. Provides accurate *positional information for large manmade* structures and hydrographic features. Information as to composition and classification of features (bridge, commercial buildings etc.) is provided in addition to vector list descriptions.

4. Integrated Database Components

In this section we will highlight some of the major components of MAPS. Much of the detail has been omitted in the interest of presenting a broad overview of the major design concepts and capabilities. Our philosophy in the system design has been to decentralize the organization of MAPS into separate processes, with each process having a particular area of expertise and communicating through well-defined data structures or files. In some cases, where a closer coupling is desired due to frequent communications, we envision using an interprocess communication mechanism (IPC) [8]. Each component is capable of stand-alone execution which facilitates incremental system development and integration. In addition, MAPS components are valuable research tools in their own right for other members of our research group. The current MAPS implementation runs on a VAX 11/780 running the UNIX operating system with 3 megabytes of memory and 700 megabytes of disk storage. Graphics display hardware includes a Grinnell frame buffer display connected directly to the VAX over a DMA interface with hardware zoom, pan, video digitizer, and tablet input. A Dunn film recorder provides 35mm slide, SX-70, and Poloroid 8x10 hard copy.

4.1. Intelligent Image Display

Window oriented display

One of the central components of the MAPS system is *BROWSE* [6], an interactive raster image display facility. *BROWSE* is a window oriented display manager which allocates and manipulates three entities: *frames*, *windows*, and *images*. A *frame* is an allocation of z-buffer space from our Grinnell frame buffer memory, which has 32 bit planes dimensioned as 512x512 pixels. *BROWSE* maintains the state of special hardware such as programmable cursors, replicated zoom and pan, and overlay memory so that it is independently available to each frame. Multiple frames can be allocated by the user, limited only by Grinnell memory. The most frequent modes of operation are allocation of four monochromatic frames of 8 bit planes per pixel each, or a single RGB frame with 8 bits per primary color and a monochromatic frame. By toggling between frames one can quickly show stereo pairs, time ordered sequences or illustrate map to image correspondence.

Window operations

Windows are dynamically created within a frame and are "*a window into*" a portion of a specified *image*. Windows are displayed within frames at positions determined by the interactive user, or by *BROWSE*. Operations on windows include: *delete, create, copy, move location, adjust position of image, expand and shrink size, raise to top,* and *zoom*. A general image windowing capability is necessary because of the mismatch between our ability to digitize images (4096x4096 pixels) and generally available raster display technology (512x512). In addition, simultaneous display of multiple windows from different images can be used in interactive stereo, time sequence and change detection tasks, and image selection from a menu of image fragments.

Symbolic naming

BROWSE allows for the symbolic assignment of names to *image* files. A standard set of *command files* are provided which isolate the naive user from the actual organization of the image file system. For example, the command "<dc1420"

```
open /vise/washdc/wgl/dc1420/1bw.img 1bw1420 map
open /vise/washdc/wgl/dc1420/2bw.img 2bw1420 map
open /vise/washdc/wgl/dc1420/4bw.img 4bw1420 map
open /vise/washdc/wgl/dc1420/8bw.img 8bw1420 map
def dc1420 8 mapped
setmap dc1420 /vise/washdc/wgl/dc1420/gmr.map
addwin dc1420 low.resolution 8bw1420
```

"opens" or assigns symbolic names *1bw1420*, *2bw1420*, etc. to an hierarchy of resolutions of the generic image *dc1420*. A frame of type *mapped* is then defined by allocating 8 bit planes of Grinnell memory and is given the name *dc1420*. Next a file containing a lookup table mapping function which is specialized to the spectral characteristics of this image is associated with the frame. Finally, a window into the image 8bw1420 is defined in frame *dc1420* and given the symbolic name *low.resolution*. Symbolic names are consistently used throughout *BROWSE* to describe *frames*, *windows* and *images*. The command interpreter allows for unique substring abbreviations for any symbolic name, and redirection of input from a command file (as shown above) instead of the user's terminal.

Image resolutions

The relationship among resolutions is often explicitly used by *BROWSE* commands, which perform image zooming, image placement, or prompt for an image coordinate. The general paradigm is to have the user specify an area of interest in a low resolution which allows for maximum amount of image context to be displayed. Once the area of interest is specified, *BROWSE* automatically creates a new high-resolution display window. The user performs operations which require high resolution, such as feature identification and segmentation, image registration, query specification, in this window. The ability to rapidly present both the entire scene at reduced detail and selected portions of the image at higher resolution is essential to interactive photo interpretation.

Photograph 1 illustrates the use of a reduced resolution window *low.resolution* to provide context for several full resolution windows, *jefferson.memorial*, *watergate.hotel*, *monument*, *white.house* and *lincoln.memorial*. This display was created using the windowing and zooming features of *BROWSE*. A partial segmentation associated with this image is displayed in color from Grinnell overlay memory. The loss of color and lack of contrast in publication may make it difficult to see the vector outlines.

4.2. Image Segmentation

In MAPS, map knowledge acquisition involves the integration of image segmentations and collateral map data. Image segmentations specify a 2D vector description of cultural and natural features. Features are classified as *point*, *linear*, or *areal* and are given identifiers which reflect their proper name (*kennedy center*) or feature type (*runway1*). Labeling of features is performed during hand segmentation, or as a post-processing operation on machine generated descriptions. Image segmentation files can be generated by a combination of the following three procedures:

Hand generated
: The user interactively specifies the position and shape of a feature using a high resolution display and a cursor. Capabilities include editing descriptions and image segmentation display in multiple levels of detail.

Map generated
: Given an image-to-map correspondence we can use existing image database segmentations to generate a segmentation for a new image. This first order approximation can be edited by hand, or processed by machine to yield a composite image description.

Machine generated
: Experimental coarse-fine segmentation using region growing and edge profile analysis has begun to be tested. Briefly, the technique is to use a coarse hand or map segmentation to specify the area within which a detailed machine segmentation should be performed. The user can incrementally accept, reject or edit descriptions as they are generated.

Figure 2 is plotted from the image segmentation file associated with aerial image in photograph 1. Symbolic names from the segmentation file have been positioned by hand. The area portrayed corresponds to the *low.resolution* window.

4.3. Landmark Selection

In order to perform image to map correspondence, we have assembled a landmark database of approximately 140 landmark features. Typical features include road intersections and traffic circles, corners of park areas, bridge access ramps, and building corners at the ground-level. Selection criteria included the following:

Uniqueness
: Easily visible from above, uniquely shaped, easily measured; e.g. ends or junctions of linear features rather than "center".

Non-temporal
: The landmark feature should not be sensitive to normal seasonal changes in foliage or water levels. This ruled out many interesting river and park landmarks having distinctive structure in aerial photography.

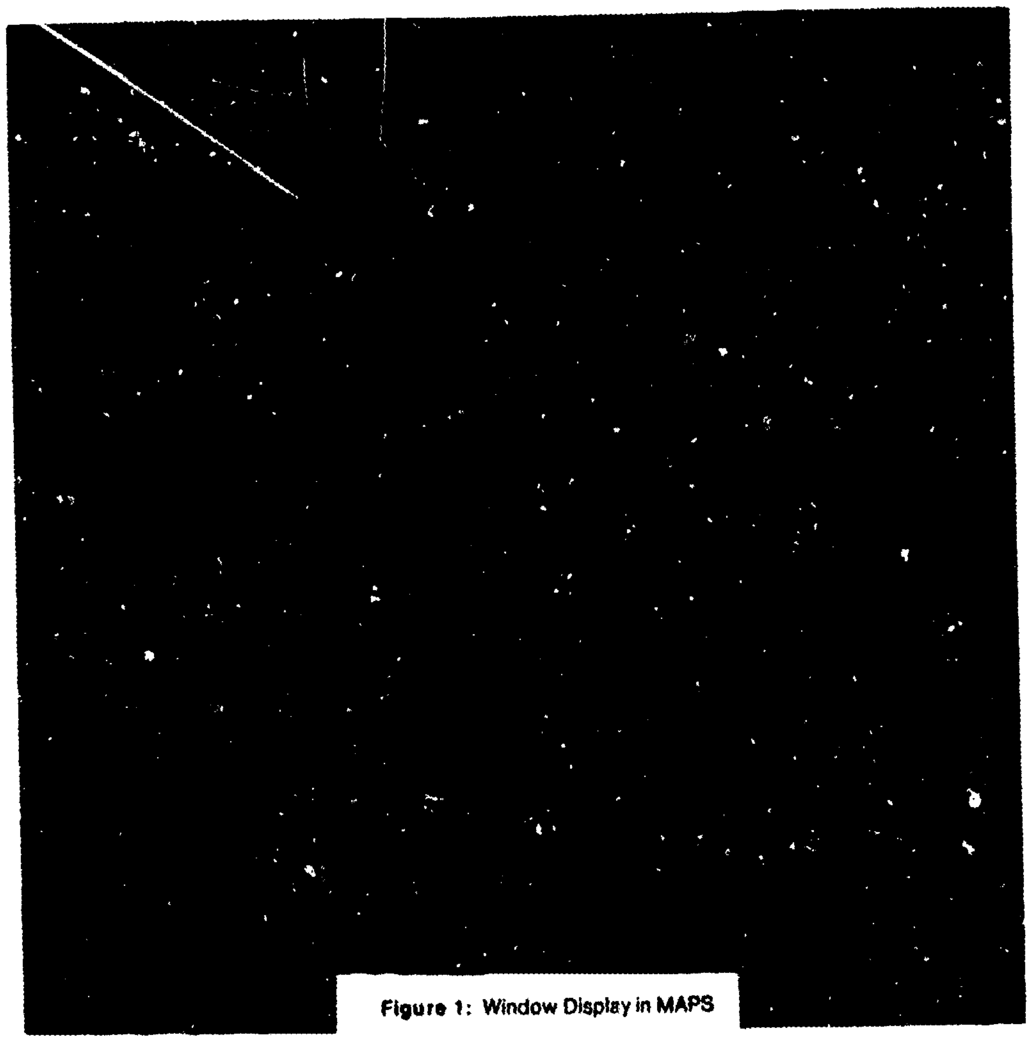


Figure 1: Window Display in MAPS

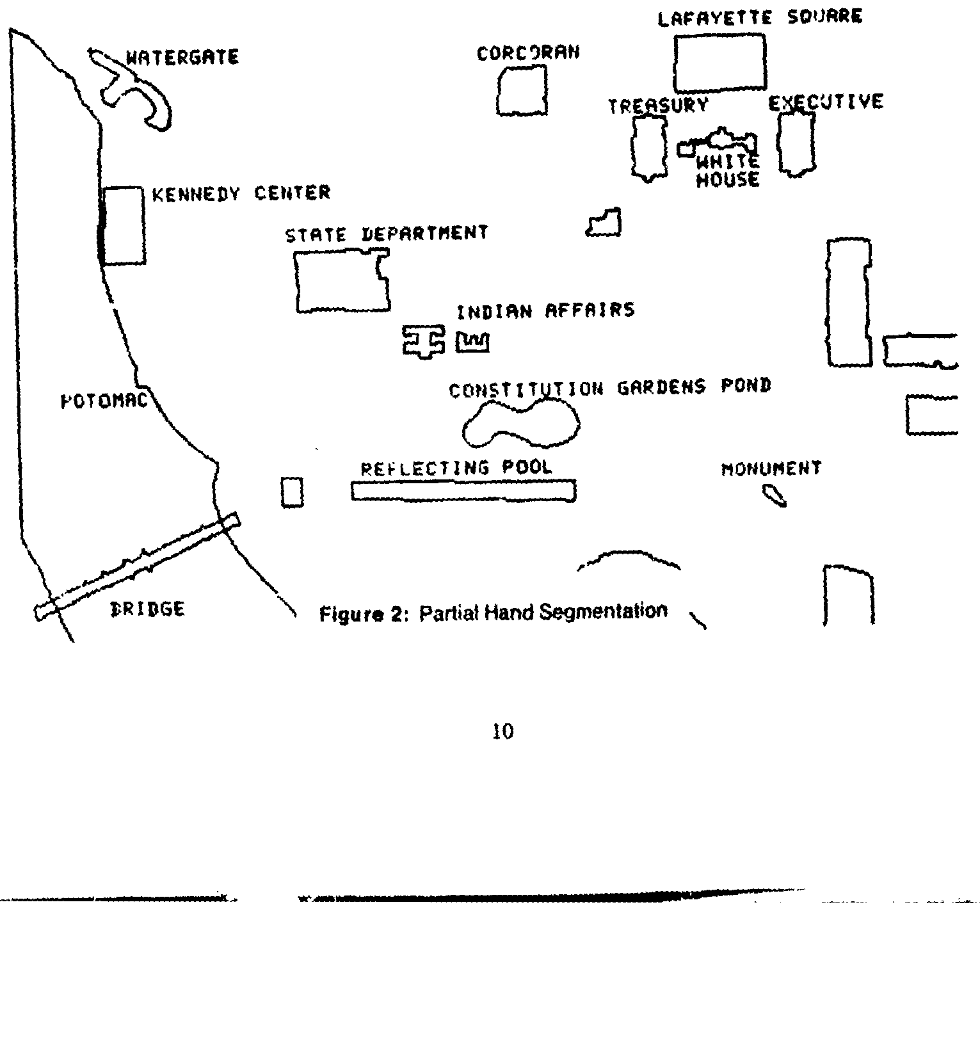


Figure 2: Partial Hand Segmentation

Spatial

Landmarks should be spatially spread in order to provide the capability to perform accurate correspondence over the entire task area.

2 Dimensional and 3 Dimensional

Initially the correspondence between our map and data base is two-dimensional. This implies that features with extreme elevations, such as the roof of multi-story buildings, are not appropriate. Corners of buildings (at ground level) are currently used. However, we expect that as the system grows, 3-d features will be accumulated and used to compute the camera model of a given picture.

Each landmark description in the MAPS database consists of map and image coordinates, a textual description of the landmark, and an image fragment containing the landmark point. Figure 3 gives a sample landmark description from the database.

The landmarks are manually created and edited using an interactive display program. The use of the landmark database component will be discussed in the following section on image-to-map correspondence.

```
LDM>    base file name  [mcphersq]
  latitude N38 54 8 500
 longitude W77 2 2 400
307,1758 in /vise/washdc/wgl/dc38617/1bw.img
landmark image at resolution 1

                mcpherson square
A small park square in District of Columbia NW.
Located just north of Veterans Administration
and Lafayette Buildings.  Control point is in
upper right corner of the landmark image.
```

Figure 3: Landmark description for McPherson Square

4.4. Image-to-Map Correspondence

In order to integrate image segmentations, map image data, and collateral map data such as USGS and DMA feature and terrain descriptions into a consistent representation we perform an image-to-map correspondence. The steps in performing the correspondence between a new image and the map database are as follows:

Step 1. Initial identification

An initial set of corresponding points are specified by selecting descriptions from the landmark database. Landmark selection is performed by explicit naming or menu selection of landmark image fragments. We plan to extend this to selection by landmark description, e.g. "bridge over Potomac", to display image fragments satisfying the description. Once a landmark is selected, users are prompted to graphically indicate the corresponding point in the new image. The selected landmark image fragment is displayed for reference. If the user specifies correspondence in a low resolution window, a high-resolution window is automatically created and the point is respecified with greater precision. An initial guess of map coverage can be computed after specification of the first corresponding point using image scale, digitization size, and assuming image orientation. This estimate allows MAPS to generate image fragments of plausible landmarks from the outset, in lieu of user expertise or familiarity with the region covered by the image.

Step 2. Perform correspondence

Our correspondence process [3] is invoked with a file of image pixel coordinates and corresponding landmark latitude/longitude pairs. This *correspondence pair file* is used by MAPS to produce coefficients for image-to-map *linear*, *second order* and *third order* polynomial approximations. Mean error, variance for each point, and a measure of model error (Predicted Sum of Square (PSS)) are computed. The *best order* model is evaluated based on these measures. Correspondence *coefficient files* are associated with each image in the database.

Step 3. Landmark candidate generation

The coefficient file generated in step 2 is used to calculate the map coordinate of the four image corner points. These coordinates are used to search the landmark database for possible new landmarks within the image coverage.

Step 4. Select, Add, Modify, Delete

The image-map correspondence pairs used to generate the coefficient file in step 2 are redisplayed. Bounding boxes indicating the original point selected by the user, and the position calculated by application of map-to-image coefficients are superimposed over the image display. Landmark candidates generated in step 3 are also displayed. The user can select a particular landmark, zoom the image to view a high-resolution window, and modify or delete correspondence pairs. Landmark candidates may also be selected for addition to the correspondence pair file.

Step 5. Iterate

The addition, deletion and modification of correspondence pairs can continue by invoking the correspondence process (step 2) as desired. Error statistics generated by the correspondence routines are useful for detecting stopping conditions.

Photograph 4 shows a sample correspondence display. The position of the overlapping rectangles in the image (color overlays) indicates to the user the magnitude of the local error for each of the correspondence pairs. The *mcpherson square* landmark previously described in figure 3 is zoomed in *cwindow*. Future directions in the area of image-to-map correspondence include using multiple points in specification of landmark position, using local 3D scene information, and automating the selection of landmark and image correspondence pairs.

4.5. Map Database

The *map database* component of MAPS is central to providing access to imagery, guiding photo interpretation, and processing queries about manmade and natural features. Through the image-to-map correspondence process, map knowledge can be applied to any image, and the spatial relationships of sets of imagery can be established. We have described how image segmentations are

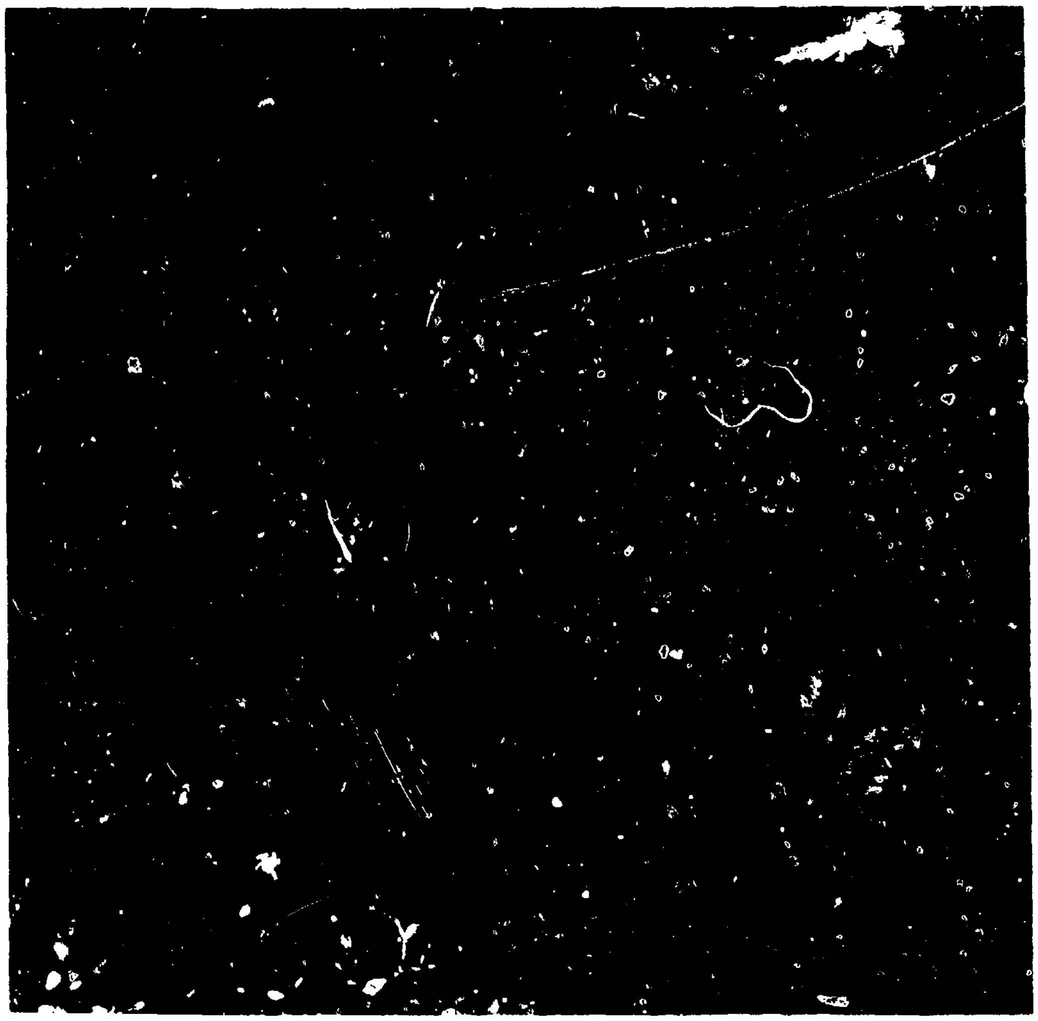


Figure 4: Image to-Map Correspondence

used to extract feature descriptions and how these segmentations can be integrated through the same correspondence process. Similarly, feature descriptions from collateral data, such as the DMA radar simulation database, are included in MAPS.

We would like to extend the notion of a purely feature-oriented map database to one which has the ability to represent general spatial knowledge in the scene domain. This *conceptual map* provides a framework within which individual map features can be associated with high-level semantic map descriptions. Conceptual maps capture the spatial arrangement in urban areas of neighborhoods, political, and geographical boundaries. For example, terms such as "Northwest Washington", "Foggy Bottom", "Alexandria, Virginia" are often used to describe general areas within and around Washington D.C. They provide an important mechanism for symbolic access into an image database, e.g. "display images of Georgetown later than 1976". However, depicting precise boundaries of conceptual features from aerial imagery is a difficult problem. In many cases boundaries are ill-defined and highly dependent on the user's own spatial model. There is clearly a hierarchy corresponding to *levels of detail* among conceptual features which must be preserved in order use such knowledge effectively.

The advantages of providing such a representation are important. First, conceptual features can be used to partition the map feature space. This partitioning would be based on natural spatial relationships, ones which are likely to arise in database queries, rather than artificial cellular or raster decompositions. Second, many queries into the map database can be resolved at the symbolic level, without resorting to geometric computations. This is particularly true if static relationships such as *intersection*, *inclusion*, and *adjacency* can be pre-computed and represented in the conceptual map. Queries of the form "does x intersect y" should be handled by looking up the binary relationship *intersect* for x, y and all entities which make up x and y. Of course, when the actual location of the intersection is required, a geometric operation must be performed. However, if x and y are hierarchically represented by their conceptual components, a symbolic query can be used to find the components which intersect, and perform intersection computations locally.

We are currently beginning work on conceptual map representations for features such as *roads*, *buildings*, and *bridges*. Concurrently we are exploring the issue of spatial containment and hope to demonstrate symbolic image access using conceptual maps within MAPS.

5. Interface to Image Processing Techniques

Given a system with the capabilities of MAPS, we believe the prospects for applying current image processing technology to detailed urban city scenes improve markedly. Feature or landmark extraction can be guided by spatial and structural constraints. Even errorful map descriptions can allow for the localization of processing. Given an assumption of consistency in error, one can compensate by relaxing the predicted position of an object in the scene. Such pruning of the "search space" can enhance the usefulness of local image segmentation techniques which focus processing in the local area with approximately known properties. Examples of current local techniques include edge profile analysis [2], local image registration techniques [4], edge linking by directional analysis [7], and hough transforms [9].

Structural constraints in the form of shape knowledge can be used to fill in missing or noisy data. Complex aerial and satellite imagery contain many examples of building surfaces in shadow or completely occluded by other buildings, or roads obscured by tree cover. Besides these inherent 2D image from 3D scene problems, image formation and illumination conditions often make it difficult for even humans to identify boundaries between features. However, when structural constraints such as rectangular shape or direction of gravity are given, the application of gradient space surface orientation theories [1]can be made tractable for the scene under analysis.

6. Conclusions

In this paper we have described the major components of the MAPS system: intelligent image display, image segmentation, landmark selection, image-to-map correspondence, and the map database. The use of spatial and structural constraints in the interpretation task has been our main focus. We are continuing to incrementally develop our map representation for the Washington D.C. area.

7. Acknowledgements

Programming assistance by Jerry Denlinger, Fuminobu Komura, and John Zsarnay contributed to many of the MAPS system components.

8. Bibliography

[1] Kanade, T. and Kender, J.R.
Mapping Image Properties into Shape Constraints: Skewed Symmetry and Affine-Transformable Patterns.
In *Proceedings of the Workshop on Picture Data Description and Management*, pages 130-135. IEEE, August, 1980.

[2] Kanade, T.
Recovery of the Three-Dimensional Shape of an Object from a Single View.
Artificial Intelligence (to appear) 14, 1981.

[3] Komura, F. and McKeown, D.M.
Interactive Correspondence in MAPS.
Image Understanding Group Memo 12, Carnegie-Mellon University, Pittsburgh, PA., 1981.

[4] Lucas, B. L. and Kanade, T.
An Iterative Image Registration Technique with an Application to Stereo Vision.
(in this volume).

[5] McKeown, D.M.
Knowledge Structuring in Task Oriented Image Databases.
In *Proceedings of the IEEE Workshop on Picture Data Description and Management*, pages 145-151. Asilomar, Cal., August, 1980.

[6] McKeown, D.M., Denlinger, J.L.
BROWSE: A Generalized Image Display Facility.
Image Understanding Group Memo 6, Carnegie-Mellon University, Pittsburgh, PA., 1980.

[7] Nevatia, R. and Babu, K.
Linear Feature Extraction and Description.
In *Proc. IJCAI-79*, pages 639-641. August, 1979.

[8] Richard F. Rashid.
An Inter-Process Communication Facility for UNIX.
Technical Report, Carnegie-Mellon University, Pittsburgh, PA., 1980.

[9] Sloan, K., and Ballard, D.
Experience with the Generalized Hough Transform.
In *Proceedings: DARPA Image Understanding Workshop*, pages 150-156. April, 1980.

DETERMINING THE INSTANTANEOUS DIRECTION OF MOTION FROM OPTICAL FLOW GENERATED BY A CURVILINEARLY MOVING OBSERVER

K. Prazdny

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

A method is described capable of decomposing the optical flow into its rotational and translational components. The translational component is extracted implicitlyby locating the focus of expansion associated with the translational component of the relative motion. The method is simple, relying on minimizing an (error) function of 3 parameters. As such, it can also be applied, without modification, in the case of noisy input information. Unlike the previous attempts at interpreting optical flow to obtain elements, the method uses only relationships between quantities on the projection plane. No 3D geometry is involved. Also outlined is a possible use of the method for the extraction of that part of the optical flow containing information about relative depth directly from the image intensity values, without extracting the "retinal" velocity vectors.

INTRODUCTION

The distribution of velocities on the projection surface arising as a consequence of the relative motion of objects with respect to the observer, the optical flow (Gibson, 1950, 1955), contains information not only about the (relative) motion itself, but also about the three-dimensional disposition of the set of texture points of which a given set of image elements is a projection <Note 1>. The distribution of image velocities on the projection surface is a function of three parameters: the (relative) motion of objects, their distance to the center of projection, and the local three-dimensional geometry of the objects. Fortunately, however, their effects are, conceptually at least, separable (Prazdny, 1981).

The problem of extracting information from optical flow can conveniently be divided into two stages. First, one has to develop a method of extracting the image velocities from changes in the image intensity values on the projection surfaces. Following this, it remains to solve the problem of computing the required information about (local) surface orientation, relative depth and motion from the distribution of these velocities. While the interpretation of optical flows logically depends on the solution to the problem of extracting the constituent velocities, the problem can and should be studied on its own, for its solution not only explicitly voices the requirements on the quality of the velocity extraction process, but also determines the ultimate success of the whole enterprise.

Recently, optical flows (and time varying imagery in general) have received growing attention among the computer vision community as a source of possible information about a scene. Nakayama and Loomis (1974) and Fennema and Thompson (1979) studied how discontinuities in the "retinal" velocity field could be used for segmentation purposes. Clocksin (1980), Gibson et al. (1955), and Lee (1974) studied how the optical flow generated by an observer translating in a stationary world provides information about (local) surface orientation. Koenderink and van Doorn (1976), Longuet-Higgins and Prazdny (1980), and Prazdny (1980, 1981), studied the extraction of surface orientation, (relative) depth and motion from optical flow generated by an arbitrary curvilinear motion. Another kind of approach, sometimes considered more suitable for a computer vision system, relies on interpreting a sequence of static images as discrete snapshots (see e.g., Nagel, 1981; Aggarwal and Badler, 1980). The computation of "retinal" velocities from image intensity values was studied by Fennema and Thompson (1979), Hadani et al. (1980).

and Horn and Schunck(1980). Similar approaches based on matching various higher-order image structures obtained from two (temporally) consecutive images were attempted by, for example, Barnard and Thompson (1980).

In this paper, we outline a method for obtaining the instantaneous direction of (relative) motion from optical flow manifested as image motions on the planar projection surface. We use polar projection as the model of the physical image forming process. Also, we assume throughout the paper that our world contains only rigid and opaque objects. The method presented here does not use projective or geometrical relations, as might be expected from the use of the polar projection; it is based on computations and relationships defined and measurable solely on the projection plane. For example, the method does not require knowing the visual direction (a 3D vector) of a "retinal" point (as was required, e.g., in Prazdny, 1980).

Before outlining the method, we briefly consider a few relevant facts. Optical flow can be (instantaneously) decomposed into two independent components (Koenderink and van Doorn, 1976; Nakayama and Loomis, 1974), a rotational and a translational component. However, only the translational "retinal" field consists of motion along straight lines all intersecting at a common point, the focus of expansion (FOE). This point corresponds to the point where the (three dimensional) vector tangent to the motion path described by O at a given instant) pierces the projection plane. Our method, by searching for this point, effectively decomposes the optical flow field into its two constituent fields. Briefly, the method is based on minimizing an error function of three parameters. The construction of the function reflects the following observation: if the three parameters specifying the rotational component of the (relative) motion are chosen properly, the translational "retinal" field yield lines all meeting at FOE. We do not require the spatial derivatives of the "retinal" velocity field, as in Koenderink and van Doorn (1976) or Longuet-Higgins and Prazdny (1980). The method can, but does not have to be, implemented as a local computation. While we have chosen, for simplicity, to consider only the case of an observer moving in a stationary world, it should be noted that the method has a much more far-reaching implication. Because it produces a description of relative motion, it can be applied to a region of the image locally to describe the (relative) motion of that region independently.

LOCATING THE FOCUS OF EXPANSION (FOE)

To see that it is possible to decompose the instantaneous positional velocity field on the projection plane into its two components, consider the effects of rotation and translation separately. It is advantageous to imagine that the optical flow field is generated by the motion of the observer in a stationary environment. This conceptualization has an immediate interpretation and is, of course, legitimate, for all motion considered here is relative.

Consider the observer rotation first. Because the rotational component of the relative motion does not carry information about the 3D disposition of the texture elements, the motion of an image element on the projection plane will depend only on its position on PP. A rotation vector (angular velocity vector perpendicular to the instantaneous plane of rotation) can be decomposed into two components, one parallel to the projection plane (PP), and one perpendicular to it (see Figure 1) <Note 2>.

Consider the rotation about the vector perpendicular to PP first (the z-axis in Figure 1). For each "retinal" point P with coordinates (x,y), the rotation of the observer about the z-axis (perpendicular to PP) results in P moving along a circular trajectory on PP. The motion of P on PP is specified by a direction vector $\overline{c}=(-y,x)/\sqrt{(x^2+y^2)}$, and by the magnitude $c=c_0\sqrt{(x^2+y^2)}$, where c_0 is the speed of a "retinal" point at a unit distance from O' (the center of the "retinal" coordinate frame). The (2D) velocity of P(x,y) due to observer rotation about the z-axis is thus given by

$$(1)\quad \underline{c}(x,y)=c_0(-y,x)$$

Consider now the situation in which the observer rotates about a vector parallel to the projection plane <Note 3>. To simplify the discussion, we consider only rotation about an axis (through O) parallel to the "retinal" y-axis. The expression for rotation about a parallel to the x-axis is symmetrical in the coordinates x and y [compare equations (5) and (6)]. We first show that the path of a point P(x,y) under rotation of the observer about the y-axis is a hyperbola, and then derive the expression for the velocity vector $\underline{h}_H$ at P(x,y).

Consider Figure 2. A stationary texture element in the 3D environment projects into a point P(x,y) on PP. As PP rotates about a line parallel to the y-axis, the coordinates of P will eventually become $P(0,y_0)$. Observe that the projecting ray defines a fixed visual angle with respect to the plane of rotation. It is clear from Figure 2 that

$$y_0^2 = \tan^2\epsilon = \frac{y^2}{x^2+1}$$

This is because the distance $|OO'|=1$, by assumption (this effectively scales the whole projection system by the focal distance). From this, we obtain

$$(2) \quad \frac{y^2}{y_0^2} - x^2 = 1$$

This is the equation of a hyperbola with center at origin O'. The direction of the velocity vector at P(x,y) is determined by the tangent line at that point. Differentiating (2), we obtain

$$y' = \frac{xy}{x^2+1} = \tan\xi$$

(see Figure 3). $\bar{h}_H$ is thus determined by $\bar{h}_H = (\cos\xi, \sin\xi)$. In terms of the ("retinal") coordinates (x,y) of P, this becomes

$$(3) \quad \bar{h}_H = \frac{1}{((x^2+1)+(xy)^2)^{1/2}} (x^2+1,\ xy)$$

To determine the magnitude of $\underline{h}_H$, consider two fixed points R and S on two rays such that at time t_0, the points coincide with points $P(0,y_0)$ and O', respectively (see Figure 3). The two rays define a visual angle ξ. Now at a time t_1 (after a rotation of PP by some angle), R projects into the point (x,y) while S projects into the point (x,0). It is evident that the two projections move so that at any time, their x-coordinates are the same. In other words, the x-components of their velocities on PP are the same. We know that the path of P is a hyperbola. It is thus sufficient to compute the horizontal velocity component and project it back onto $\bar{h}_H$ to obtain h_H, the magnitude of $\underline{h}_H$.

Consider Figure 4. If the point x moves with angular velocity h_0 (recall that $|OO'|=1$), then h_x is defined by

$$h_x = \frac{h_0\sqrt{x^2+1}}{\cos\eta}$$

But $\cos\eta = 1/\sqrt{x^2+1}$ (see Figure 4) so that $h_x = h_0(x^2+1)$. Projecting h_x back on $\bar{h}_H$ and combining the result with equation (3) we obtain

$$(5) \quad \underline{h}_H = h_{0H}(x^2+1,\ xy)$$

Here h_{0H} is the speed of a "retinal" element at O'. Analogously, the rotation of the observer about an axis (through O) parallel to the "retinal" x-axis results in the velocity vector h_V defined by

$$(6) \quad h_V = h_{0V}\ (xy,\ y^2+1)$$

The input data we are trying to interpret, the optical flows, consist of a set of vectors $\underline{v}''$ defining the positional velocity field on the projection plane. Because we are dealing with velocities, it is easy to see that

$$(7) \quad \underline{v}'' = \underline{c} + (\underline{h}_V + \underline{h}_H) + \underline{t}$$

where $\underline{t}$ is the velocity vector due to the pure translation of the observer. In other words, for each "retinal" locus (x,y), and a set of parameters (h_{0H}, c_0, h_{0V}), equation (7) defines a vector $\underline{t}$ which is a vector function of the three parameters.

As mentioned above, the property of the translational "retinal" velocity field (defined as straight lines specified by a given "retinal" locus (x,y) and the associated vector $\underline{t}$) is that all the lines intersect at one common point, the FOE (see Figure 5). This property makes it possible to define an error function which will lead to resolution of the vector field $\underline{v}''$ into its rotational and translational components. For a given distribution of $\underline{v}''$ on PP, we are searching for those values of the parameters (h_{0H}, c_0, h_{0V}) for which the set $T=\{\underline{t}_i\}$ is such that all lines L_i defined by the vector $\underline{t}_i$ and the retinal locations P_i intersect at a common point, the FOE.

One way of doing this is as follows. Consider an arbitrary "retinal" point P [with coordinates (x,y)] and a set of other (possibly neighboring) points $\{P_i\}$. The points P_i together with the vectors t_i define lines L_i which intersect the line L at points I_i (see Figure 6). Consider the lengths ℓ_i between the intersections I_i and the point P. The variance $V = \sum_i (\ell_i - \tilde{\ell})^2/n$ (where $\tilde{\ell}$ is the algebraic average of the ℓ_is) is a good measure of the dispersion of the intersections I_i. When the lines L_i all meet at FOE, V=0. To obtain the FOE, we thus simply minimize $V(h_{0H}, c_0, h_{0V})$. Note the way in which the decomposition is accomplished: a property of the translational field is here used to obtain the rotational field, resulting in both fields being obtained at the same time, by the very same computation. The method, being minimalization of a distribution measure, can also immediately be applied when the input data (the vectors $\underline{v}''$) are noisy.

SOME EXPERIMENTAL RESULTS

The schema described above was tested in a (simulated) world of planar surfaces. The results are encouraging. Eight points surrounding a central point were used to define the set $\{P_i\}$ to obtain the variance V. It should be noted that while in our implementation neighboring points were used (the neighborhood subtended about 15 degrees of arc), this is by no means a necessary condition. A direct minimalization scheme attributed to Nelder and Mead (Nash, 1979) was used to minimize the variance V. The scheme was used mainly for its simplicity and ease of encoding. The values of h_{0H}, c_0, and h_{0V} were restricted to lie between ±90 degrees of arc/sec (the "negative" values corresponding to counterclockwise rotations). This feasible region was defined to restrict the search space to meaningful values and to prevent possible divergence of the iterative process. The minimalization procedure converged to a correct solution from any initial guess within this feasible region. Not all eight distances ℓ_i were used to define V. To minimize the influence of (quantization) errors, the lengths ℓ_i were ordered in magnitude and the two extremal magnitudes were discarded. We also tried to use the range of ℓ_i (defined as $|\ell_{max} - \ell_{min}|$) as the error function with good results. In both cases, the FOE was located precisely (using single precision arithmetic [7 significant digits]). When the precision with which the vectors $\underline{v}''$ were defined was lowered to 4 significant digits (the angular error made by this quantization depends on the magnitude of the vector v'' [see Figure 7]), the FOE was located within approximately ±5 degrees of arc of the correct position. Extensive testing with real data (and using a more efficient and faster minimalization schema) should be performed to determine how the errors in $\underline{v}''$ propagate through the computations and affect the precision with which the FOE can be obtained.

DISCUSSION AND CONCLUSIONS

It is important to realize precisely what has been achieved, and how. Given a set of "retinal" vectors $\underline{v}''$ on the planar projection surface, we have shown that it is possible to extract the translational velocity field, containing all information about spatial disposition of the texture elements, solely by computations using data available on the projection surface (see Figure 8). In fact, besides the velocity vectors $\underline{v}''$, only the positions of the corresponding loci on the plane with respect to a fixed reference point (the "fovea") are required. Another feature of the method is that it can be implemented as a local computation (the radius of the neighborhood would have to be large enough), and thus performed at many "retinal" locations in parallel, thus decreasing the dependence of the method on a good initial approximation to the parameters h_{0H}, c_0, and h_{0V}. The simplicity of the method is striking, especially in comparison with other methods purporting to achieve the same results (e.g., Prazdny 1980; see also Nagel 1981). The method requires only a few points and the corresponding "retinal" velocities as input (for example, in the visual periphery, which is apparently used by the human visual system to compute egomotion [Johansson, 1977]). One disadvantage of the method is that it fails when the direction of instantaneous motion is parallel to PP. In this case, the FOE is undefined (it corresponds to an ideal point of the projective plane). This is not a serious drawback, however. Another similar method based on maximizing the parallelism between the vectors defining the translational field could take care of this situation.

It is also important to realize that once the FOE has been computed, we immediately know the direction of the translatory motion on the projection plane at each "retinal" locus; it is simply the line connecting the FOE with the given locus (on the "retinal" plane). To obtain information about (relative) depth or (local) surface orientation, we need to compute only the magnitude of motion in this direction; the two-dimensional problem is thus reduced to a more manage ble one-dimensional problem. This leads directly to a more general schema where only the velocities (v") of a few "interesting " image elements (at "prominent" locations where the velocity $\underline{v}$" can easily be detected) are computed first to locate the FOE. Following this the magnitude of the translatory motion at each image point would be computed without explicit extraction of the optical flow (the velocities $\underline{v}$") itself. As was noted by Batali and Ullman (1979) or Horn and Schunck (1980), one can compute, by a local computation, only the velocity component in the direction of the gradient of the image intensity function at a given "retinal" locus. But this is all we need if the FOE were already located: by projecting this velocity component onto the direction of the translatory field at a given image plane locus, we would obtain the (relative) depth information in its purest form - as the distribution of the magnitudes of the translatory field.

To summarize, we have shown how the direction of (relative) motion can be computed by a simple minimization computation operating on data available on the projection surface. The method can be implemented locally and is also feasible biologically. Speculatively, perhaps, its operation might be reflected in the recent finding of the "looming" or changing-size channels in the human visual pathway (Regan, Beverly, and Cynader, 1979); Beverly and Regan, 1979).

NOTES

<Note 1>

In general, only conclusions about relative quantities can be derived by interpreting optical flows. Local surface orientation, relative depth (the ratio of distances of two texture elements in two distinct visual directions), and relative motion are such quantities.

<Note 2>

The following notational convention will be used throughout the paper. $\underline{n}$ denotes a vector, $\bar{n}$ is its unit vector, and n is the norm of $\underline{n}$, i.e., $\underline{n} = n\bar{n}$. Angular velocities are conceptualized as axial vectors, i.e., vectors perpendicular to the instantaneous plane of rotation, with magnitude equivalent to the angular speed. The word "retinal" will be used to denote the projection plane, PP. P(x,y) denotes a "retinal" point P with "retinal" coordinates (x,y) in the two-dimensional coordinate frame centered at O'.

<Note 3>

The set of paths traced by the image elements under this motion is a family of hyerbolas with principal axes inclined at angle ω with respect the y-axis. The family is symmetrical about a straight line through O'. This is the line corresponding to the intersection of the plane of rotation with the projection plane.

REFERENCES

Aggarwal, J.K. and Badler, N.I. (eds.) 1980, Special issue on motion and time-varying imagery, IEEE Trans. Pattern Analysis Machine Intelligence 2,6.

Barnard, S.T. and Thompson, W.B. 1980, Disparity analysis of images, IEEE Trans. Pattern Analysis Machine Intelligence 2,4, 333-340.

Batali, J. and Ullman, S. 1979, Motion detection and analysis, DARPA Image Understanding Workshop, 69-75, Science Applications, Inc., Arlington, VA.

Beverley, K. and Regan, D. 1979, Separable after-effects of changing size and motion in depth, Vis. Res. 19, 727-732.

Clocksin, W.F. 1980, Perception of surface slant and edge labels from optical flow, Perception 9, 253-269.

Fennema, C.L. and Thompson, W.B. 1979, Velocity determination in scenes containing several moving objects, Computer Graphics Image Processing 9, 301-315.

Gibson, J.J. Olum, P. and Rosenblatt, F. 1955, Parallax and perspective during aircraft landing, Am. J. Psych. 68, 372-385.

Hadani, I. Ishai, G. and Gur, M. 1980, Visual stability and space perception in monocular vision, J. Opt. Soc. Am. 70, 60-65.

Johansson, G. 1977, Studies in visual perception of locomotion, Perception 6, 365-376.

Koenderink, J.J. and van Doorn, A.J. 1976, Local structure of movement parallax of the plane, J. Opt. Soc. Am. 66, 717-723.

Lee, D.N. 1974, Visual information during locomotion, in: Perception: Essays in Honor of J.J. Gibson eds. McLeod, R.B. and Pick, H.L., Ithaca, NY: Cornell University Press.

Longuet-Higgins C. and Prazdny, K. 1980, The interpretation of a moving retinal image, Proc. R. Soc. London B-208, 385-397.

Nagel, H.H. 1981, On the derivation of 3D rigid point configurations from image sequences, IEEE Conf. Pattern Recognition Image Processing (to appear).

Nakayama, K. and Loomis, J.M. 1974, Optical velocity patterns, velocity sensitive neurons and space perception, Perception 3, 63-80.

Nash, J.C. 1979, Compact Numerical Methods for Computers, New York: Wiley.

Prazdny, K. 1980, Egomotion and relative depth map from optical flow, Biological Cybernetics 36, 87-102.

Prazdny, K. 1981, Relative depth and local surface orientation from image motions, Technical Report TR-996, Computer Science Center, University of Maryland, College Park, MD 20742.

Regan, D., Beverley, K., and Cynader, M. 1979, The visual perception of motion in depth, Sci. Am. 241, 136-151.

ACKNOWLEDGEMENTS

The support of the Defense Advance Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged.

The author also thanks Professor Azriel Rosenfeld for his useful comments.

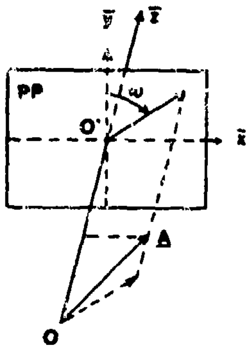


Figure 1. Without loss of generality, the projection plane PP can be positioned at unit distance from the center of projection O, and parallel to the yz-plane. Any vector A can then be decomposed into a component parallel to the projection plane, and a component perpendicular to PP. O' is the center of the "retinal" coordinate frame (2D).

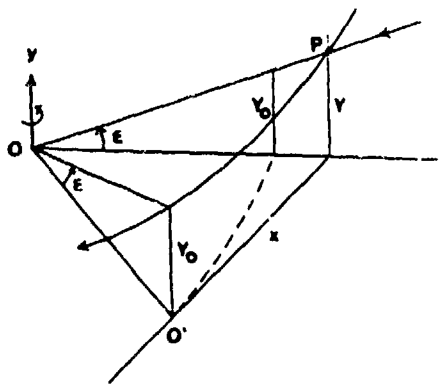


Figure 2. When the observer rotates about the vector y, the path described by an image element P on the image plane PP is a hyperbola.

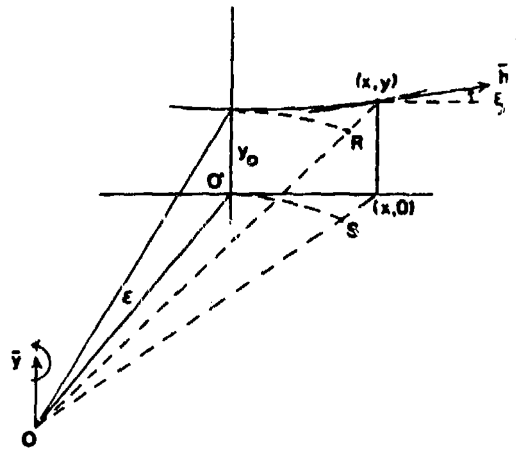


Figure 3. The observer rotates about the line parallel to $\bar{y}$ through O. The direction of the velocity vector at P(x,y) is determined by tanε. The projections of the points R and S on PP have the same horizontal velocity components.

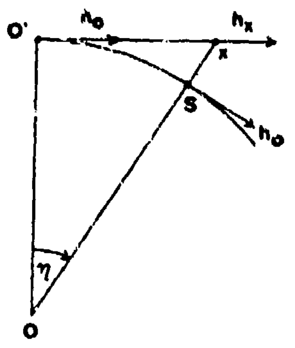


Figure 4. The angular speed of a point x is

$$h_0 = \frac{h_x \sin(\pi/2 + \eta)}{\sqrt{x^2+1}}, \text{ by}$$

definition ($|OO'| = 1$). From this, h_x can be computed directly as a function of the parameter h_0.

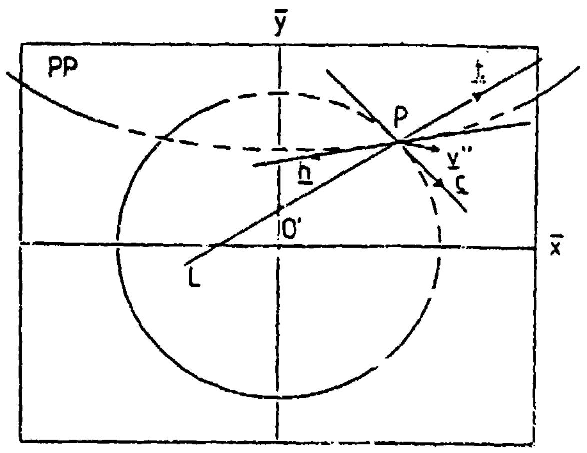


Figure 5. An image velocity v'' (on the planar projection surface PP) of a point P can be resolved into three components. The hyperbolic component h is due to the rotation of the ray about an axis (through O) in the projection plane (the angular velocity is a linear combination of x and y). The circular component c is due to the rotation of the ray about an axis (through O) parallel to z. The translational component t is the remaining vector which constrains the decomposition of v''; t is constrained to be such that $\forall Q_i \in PP$: (L_i intersect in one common point). In the illustration above, the direction angle of the hyperbolic field is zero, i.e., the observer rotates only about a line parallel to the y-axis.

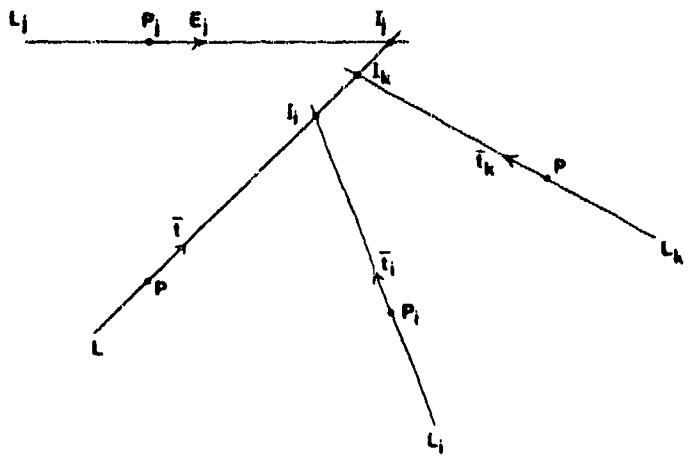


Figure 6. To find the intersection of L with L_i (I_i), we have to solve for ℓ in

$$\underline{P}+\ell\underline{t}=\underline{P}_i+\ell_i\underline{t}_i$$

To obtain ℓ we multiply both side by $\underline{t}_i'$, the perpendicular to $\underline{t}_i$. This yields

$$\ell = \frac{\underline{t}_i' \cdot (\underline{P}_i-\underline{P})}{\underline{t}\cdot\underline{t}_i}$$

where $\underline{P}$ and $\underline{P}_i$ are the (2D) position vectors on the projection plane. If $\underline{t}=(t_x,t_y)$ then $\underline{t}'=(-t_y,t_x)$.

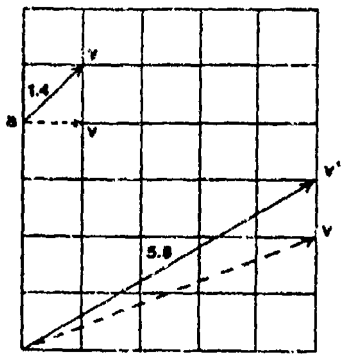
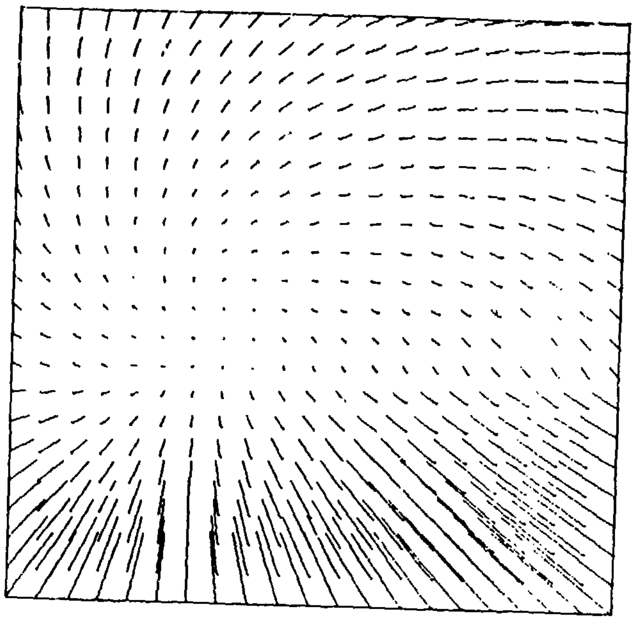
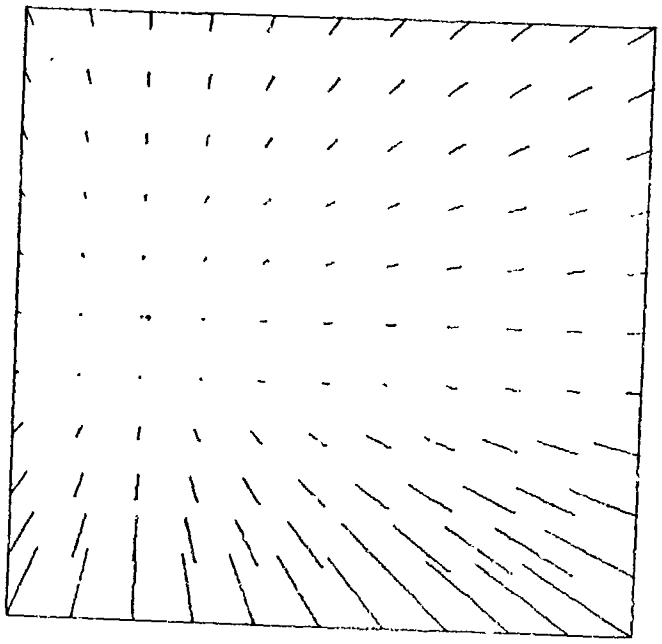


Figure 7. The quantization error increases with decreasing vector magnitude. At b, an error of about 10 degrees of arc is made by representing v as v', while at a, such a representation results in an error of 45 degrees of arc.



a. The positional velocity vector field generated by an observer "walking" backwards (on a horizontal ground plane) from a surface (plane) slanted 30 degrees (towards the observer) from the vertical.



b. Corresponding translational field. "+" denotes the FOE computed by the method (error=0). The information about the (local) surface orientation and relative depth is contained only in the magnitudes of the velocity vectors; the direction of the vectors (and the FOE) depend [illegible] on the parameters of the relative motion.

Figure 8

RELAXATION MATCHING APPLIED TO AERIAL IMAGES

K.E. Price

Image Processing Institute
University of Southern California
Los Angeles, California 90007

INTRODUCTION

We have developed a symbolic matching system which can be used for a variety of matching tasks in scene analysis. The system is designed to handle many of the problems encountered in the analysis of real scenes: noisy feature values, missing elements, extra pieces of objects, many features, many objects. At the heart of this system is a relaxation based matching scheme. A variety of relaxation procedures have been used with varying results.

The basic matching procedure [1] and the various relaxation methods [2,3] are discussed elsewhere and will only be outlined here. This paper will concentrate on a discussion of the overall matching system and the performance of the various relaxation techniques.

The input to the matching procedure is two relational structures-one for the model of the scene and the other for the input image. The structures are represented as graph structures with objects at the nodes (with associated feature values) and relations between objects as the arcs between the nodes.

SYMBOLIC MATCHING PROCEDURE

The goal of the matching procedure is to find the objects in the image (regions and lines) which best match the objects given in the model. This is essentially finding a subgraph in the image which is isomorphic to the model, except that objects may be missing and single nodes in the model may correspond to several in the image when objects are broken apart by the segmentation procedures.

The matching procedure is divided into two iterations (see Fig. 1). The outer loop consists of computing initial estimates of the assignments for all elements in the model. Up to 30 potential assignments for each element are maintained. Then a relaxation procedure is applied which updates the ratings of the assignments. When the rating for one assignment of an element in the model exceeds a threshold, the relaxation update loop is terminated and all assignments above the threshold are made permanent. Then the process continues with the recomputation of the initial estimates, but now relations (above, near, adjacent) with the permanently assigned elements can be used in the computation.

The repeated initialization is a crucial component of the process. Since the initial guesses are made only on the basis of feature value (color, shape, texture, etc.) many incorrect assignment are initially highly rated. The relaxation steps eliminate many of these mistakes, but cannot correct all of them because some correct assignments are not among the early candidates. This procedure also easily allows for multiple segments in the image to be assigned to one element in the model.

The final termination condition is the number of iterations without any assignments reaching the threshold. This number must be large enough so that valid assignments can reach the threshold but not so large that an incorrect assignment is forced, by default, to a large value. This is especially true with our primary relaxation method [1] where something is always forced to a high value (how rapid can be controlled and we use a relatively fast setting).

Several different relaxation updating schemes can be used in the inner loop. The simplest technique is the "classical" method of Rosenfeld, Hummel, and Zucker [2]. Our primary technique [1] is similar, except that the updating is always in a direction which improves a global criterion. A third method is that of Kitchen [3] which provides a different means of combining match rating from different properties. See the Appendix for a summary of the relaxation updating functions for these methods.

RESULTS

The various methods were tested on 3 different aerial views (1) a scene with 14 storage tanks, of 5 different sizes, (2) a high altitude view of the San Francisco area with 14 objects identified in the model (95 in the image), and (3) a view of the Stockton, CA. area with 20 model objects (24 or more valid matches possible, and more than 200 image elements). The results on these images allow us to make some general comments on the performance of the various methods.

The basic technique of Rosenfeld et al. [2] quickly makes several assignments in each of the test scene, but then reaches a stable state with low probabilities for the most likely assignment.

The method of Kitchen [3] allows for a variety of combining methods (by using different functions for the fuzzy set operations). Additionally, restricting the computation to use only the one most likely assignment of neighboring (related elements reduces the time substantially and improves the performance.

The results of running the various relaxation methods are presented in Table 1. The set of assignments shown n Figs. 2, 3, and 4 are the ones generated by the criteria optimization method described fully in [1] (FP in Table 1). K-1 is the Kitchen method [3] which uses MIN, MAX, and the mean for ∩, ∪, and the outer ∩ (Form 4 in his paper), the use of MIN for the outer (Form 1 in his paper) produced no assignments in our tasks. K-2 uses product instead of MIN (Form 5). K-3 is the same as K-1 exact all possible assignments of neighbors are considered rather than the one most likely assignment. Kitchen's Form 6 produced the same results on our tasks as K-2. RHZ is the classical Rosenfeld et al. method [2] and is included as a historical reference point.

The threshold for forcing a permanent assignment is 0.75 (for the Kitchen algorithms the values were normalized only for this test) and 15 iterations were run before terminating the procedure due to lack of assignments. In tests with our algorithm, changing the 0.75 threshold (between about 0.7 and 0.8) by small amounts has little or no effect. Increasing it requires more iterations and thus more time and some assignments may be lost. The results include only those assignments which exceeded the threshold (0.75) and does not include the most likely assignments at the time of termination (15 iterations). Including all these would increase the number of correct ones with no clear separation in likelihood values between correct and incorrect ones.

Several comments can be made from these results. First, Kitchen's method was not designed to fit into our matching system and is not oriented toward quickly producing unambiguous results for a few of the elements in the graph. But this feature is necessary when dealing with problem domains such as these (i.e. many feature values, many elements, similar objects, and noisy data).

Second, considering more alternatives for neighbors does not improve performance, but decreases it with a substantial increase in time. this was not totally obvious without experiments) since the likelihoods of the second, third and other alternations are much less than the most likely one they should contribute little, it any, to the computation. Tests with the other methods (K-2, FP) give similar results (decreased performance increased time).

Third, our method performs better than the others in these tasks. One major reason is the optimization procedure used to guide the updating thus some assignments are discovered early and then contribute substantially in the search for further matches. Also, in our system, all relations and features contribute to the rating rather than only the best or worst as with Kitchen using MIN and MAX.

Fourth, the 17 incorrect assignment by K-2 for Scene #1 start after 13 correct matches are located. This is accounted for by the way we handle multiple assignments for model elements and by the fact that multiple matches for image elements are only discouraged not forbidden. The change from MIN to product (between K-1 and K-2) also contributes to the problem.

The updating approach adopted by Faugeras [1] clearly performs better and operates faster. Each iteration of this program takes longer, but fewer are required.

REFERENCES

1. O.D. Faugeras, K. Price, "Semantics Description of Aerial Image Using Stochastic Labeling," IEEE-Trans PAMI, to appear, preliminary version in Proc. Image Understanding Workshop, Univ. of Maryland, April 1980.

2. A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans-SMC, Vol. 6, No. 6, pp. 420-453, June 1976.

3. L. Kitchen, "Relaxation Applied to Matching Quantitative Relational Structures," IEEE Trans. SMC Vol. 10, No. 2, pp. 76-101, Feb. 1980.

APPENDIX

We only present a summary of the equations for the relaxation updating the details are contained in the appropriate papers. Therefore many of the terms will not be fully explained. The classical Rosenfeld et al. method [2] is:

$$p_i^{(n+1)}(n_k) = \frac{p_i^{(n)}(n_k)Q_i^{(n)}(n_k)}{\sum_{n_\ell \text{ in } N} p_i^{(n)}(n_\ell)Q_i^{(n)}(n_\ell)} \qquad (1)$$

$p_i(n_k)$ is the likelihood of assignment for unit i to name k, Q is a measure of the compatibility of the assignment with assignments of neighboring units. N is the set of all possible names (image elements).

The Kitchen updating method [3] is (rewritten for our problem)

$$p_i^{(n+1)}(n_k) = LC_i(n_k) \qquad (2)$$

$$LC_i(n_k) = \left[\bigcap_{\substack{\text{all relations rel}\\ \text{s.t. } u_i \text{ rel } u_j}} \left[\bigcup_{\substack{\text{all } n_\ell \\ \text{s.t.} \\ n_k \text{ rel } n_\ell}} \left[p_i(n_k) \cap p_j(n_\ell) \cap c(u_i, n_k, u_j, n_\ell) \right] \right] \right] \tag{3}$$

$$\cap \quad = \left[\bigcap_{\substack{\text{all features}\\ \text{of } u_i}} \left[p_i(n_k) \cap f(u_i, n_k) \right] \right]$$

where and are the fuzzy logic AND and OR operations. C is a local consistency measure which is also used to compute the Q's in the equations above and below. The generality of the original formulation is not retained - only features of single objects and relations between two objects are given - not the arbitrary number of the original.

The third method of Faugeras [1] is:

$$\vec{p}_i^{(n+1)} = \vec{p}_i^{(n)} + \rho_n P_i(\vec{g}_i^{(n)}) \tag{4}$$

where ρ_n is a positive number (step size) to control the speed, P_i is a projection operator to maintain the constraint that $\vec{p}_i$ is a probability vector, and $\vec{g}_i$ is a gradient function computed from the compatibility measures and current probability values for an object and its neighbors.

		Number			
Method	Scene	Correct	Incorrect	Not Assigned	Time
K-1	1 (Tank Farm)	12	0	2	5
K-2	1	14	17	0	30
K-3	1	0	0	14	-
RHZ	1	14	28	0	29
FP	1	14	0	0	4
K-1	2 (San Francisco)	7	0	7	19
K-2	2	9	0	5	28
K-3	2	7	0	7	4:50
RHZ	2	8	11	6	53
FP	2	14	0	0	9
K-1	3 (Stockton)	9	0	11	1:25
K-2	3	16	4	5	3:00+
K-3	3	6	0	14	6:00+
RHZ	3	16	11	6	2:30+
FP	3	24	1	0	1:00

Table 1. Relaxation results.

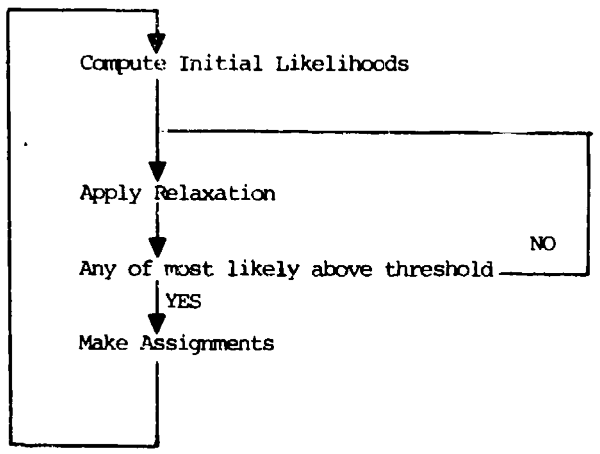

Compute Initial Likelihoods
Apply Relaxation
Any of most likely above threshold
NO
YES
Make Assignments

Fig. 1. Use of the relaxation procedure in the overall matching system.

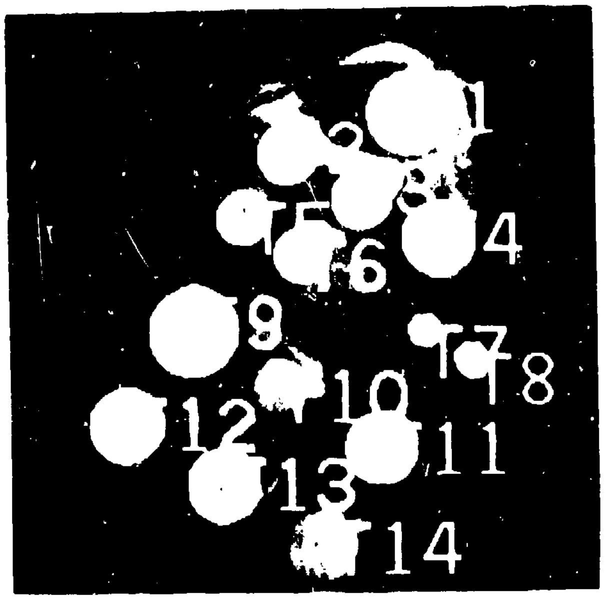


Fig. 2. Storage tank results.

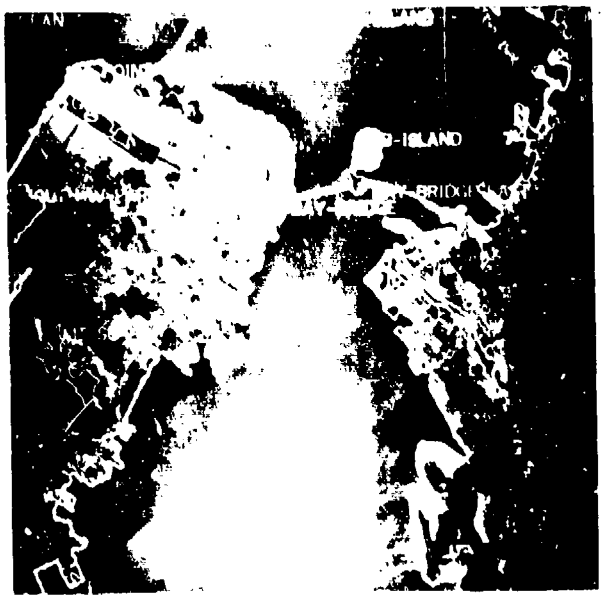


Fig. 3. San Fracisco scene results.

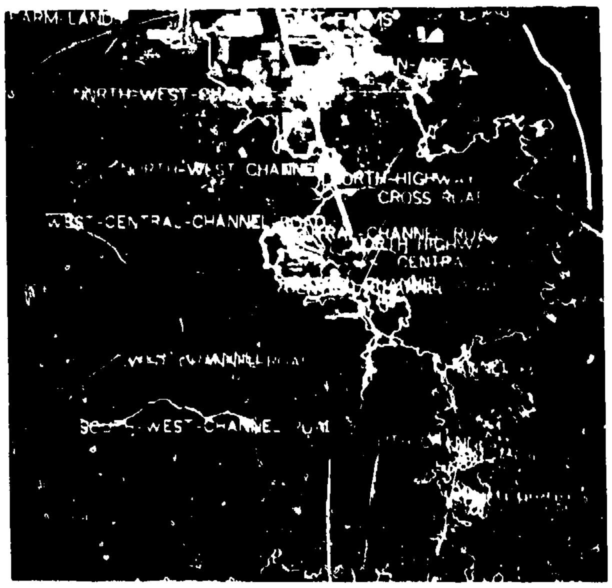


Fig. 4. Stockton area results.

LINE FINDING WITH SUBPIXEL PRECISION

P.J. MacVicar-Whelan and T.O. Binford

Artificial Intelligence Laboratory, Stanford University, Stanford, California 94305

Abstract

An intermediate level vision system that utilises grey scale levels (typically 8 bits or 256 levels in our case) has been implemented which locates and links intensity discontinuities in a digitized image to subpixel precision. The discontinuities are located and localised by utilizing the zero crossings in the laterally inhibited image of the digitized picture.

Introduction

As has recently been pointed out in earlier work in the literature (Nevatia & Babu 1978), the effectiveness of many machine vision systems is often limited by the low level processing that constitutes the first stage of the system. Typically, this stage consists of operations such as edge detection, thinning, thresholding, and linking-in other words- line finding.

Given this current state of the art and the inspiration of earlier MIT work (Binford-Horn 1972, Binford 1970, Herskovitz & Binford 1970), we have undertaken the task of seeking an improvement to this stage of vision systems. The processing reported here, a simplification of the Binford-Horn approach, differs markedly from most systems reported in the literature but has similarities to some current MIT systems (Marr & Hildreth 1979, Grimson 1980). Like the MIT systems of Binford-Horn and Marr-Hildreth, our system first applies lateral inhibition to the image, then follows this step by detecting significant signal from gradient magnitude (contrast), and finally localizes the step by zero-crossing of the lateral inhibition signal.

Operators such as Nevatia and Babu are essentially gradient operators. The gradient has a broad maximum at an edge. Such operators require a thinning process or a process of maximum selection, with degraded resolution. Thresholding, too, has degraded resolution from that demonstrated here (Binford 81). Intensities are weighted averages over a pixel area. That introduces a spatial smearing from the discrete sampling. The gradient has a maximum which cannot be located closer than the nearest integer pixel,0.29 pixel rms error, and with effects of thinning, can have worse accuracy. The lateral inhibition signal, equivalent to a second derivative, has a zero at the step, which can be interpolated through zero with an accuracy which depends on the signal to noise ratio.

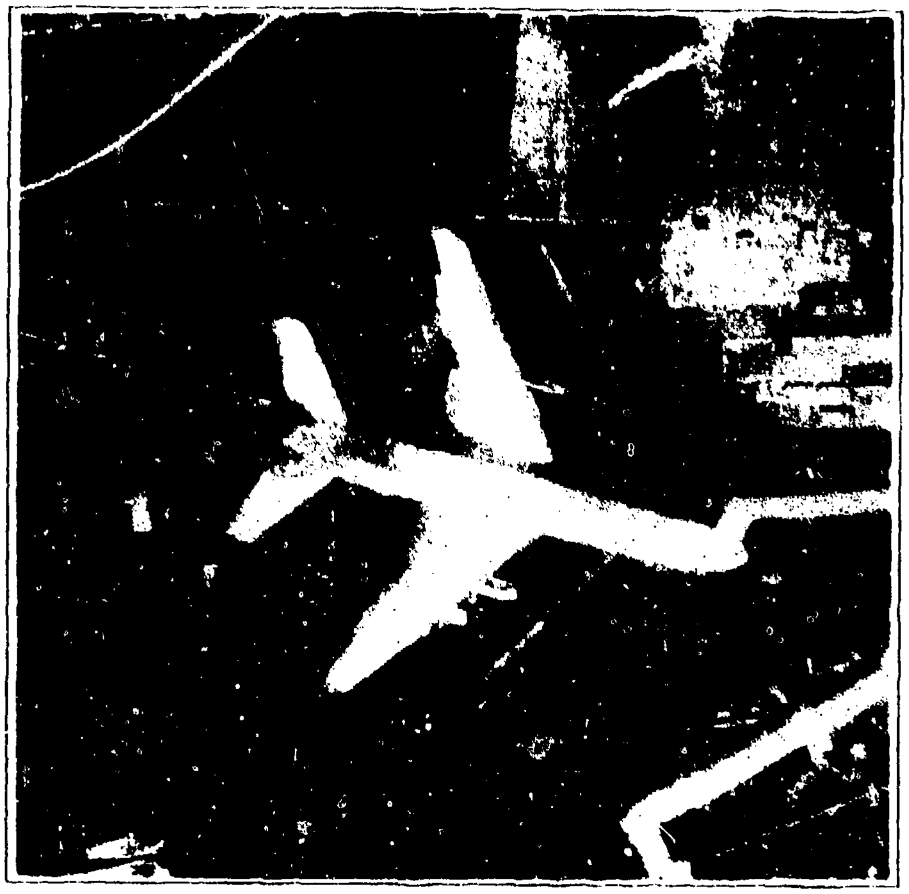
The significance of greater precision is that it makes full use of the inherent information of the signal. In cases in which thresholding or gradient operators might

Lateral Inhibition

Like Binford-Horn but unlike the MIT approach, we convolve the image with a square mask of side $2n+1$ to yield the local average intensity (over the $2n+1$ x $2n+1$ area) and subtract this average from the intensity of the pixel at the centre of the square. The resulting intensity is the laterally inhibited value of the image at the central pixel. An example of this is presented in Figure 2 using as input the image shown in Figure 1. It is this value that is used for location of the zero crossings. In the process of calculating the laterally inhibited image, linear intensity functions (including the special case of the constant function) are mapped into the zero value. An intensity discontinuity is characterised by values that rise/fall to a maximum over n pixels, a switch to a maximum of the opposite sign, and a fall/rise to zero again over n pixels.

Although it is possible to convolve the image with masks that utilise more than a central pixel, say a 2x2 or a 3x3 central window, from which the local average (the $2n+k$ average) is subtracted, we have limited our investigation to the use of a single central pixel. It yields a good result for the cases studied to date.

As is to be expected, when two discontinuities are separated by less than the mask dimension there will be interference which will result in locational errors. While



A Lockheed L1011 parked at the San Francisco Airport
Figure 1

a small mask will minimize this effect, it will be more sensitive to noise and the errors induced by it.

Discontinuity Point Detection

It is the zero crossing that occurs during the switch from one maxima to the other that is the zero crossing of interest since this zero corresponds to the location of the discontinuity.

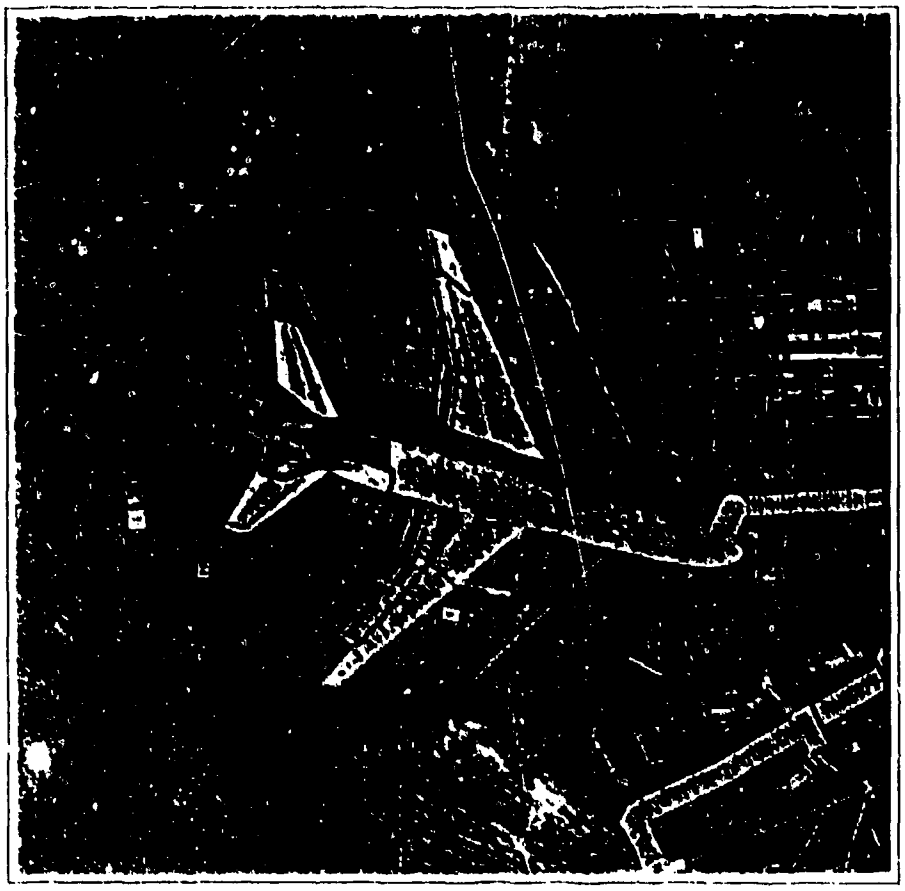
For the case of subtracting the value of the single central pixel from the local average, this switch from peak to peak occurs over a 2 pixel interval. It is the location of the zero of this switch that is the zero crossing of interest.

The exact position of the crossing was taken to be the linearly interpolated position between the pixels obtained using the values of intensity on either side of the crossing.

Zero point crossings were calculated in both the horizontal and vertical directions for the processed image.

Discontinuity Point Linking

The zero points or zero crossings of the laterally inhibited image were linked together over a mesh of dimension equal to a single pixel separation. Decisions on the linking were based upon the intensity values of the corners of the mesh which were four pixel values. Use of measured picture noise as a threshold to reject spurious crossing was accompanied by the simultaneous rejection of intensity continuities that marked the edges of shadows. In an effort to overcome this problem and improve rejection of satellite crossings resulting from the lateral inhibition process a structure filter was added. This filter examines the twelve intensity points around the four of the central mesh and rejects zero crossings that are of extent less than four pixels. The filter did not pass points unless they belonged to trajectories that exited from the 3 x 3 pixel area containing the central mesh.



The result of applying lateral inhibition to Figure 1
Figure 2

A zero crossing was deemed to occur along the edge of a mesh if one corner of the mesh edge was above zero and the other corner was below zero these 2 corner values were then used to calculate location along the edge of the crossing. Intramesh points were established using the average of the four corner points as the value at the center of the mesh. The position of the zero crossing was then calculated to lie between the centre point and one of the corner points.

Using the fact that a zero crossing point is one of a continuous trajectory of such points, the decision as to which points must be joined can easily be determined. If it is assumed that the calculated value for the center of the mesh is accurate at least in so far as to whether or not it is above or below the zero value, this information can be used to separate two trajectories that pass through a single pixel. An example of this is to be found in Figure 5.

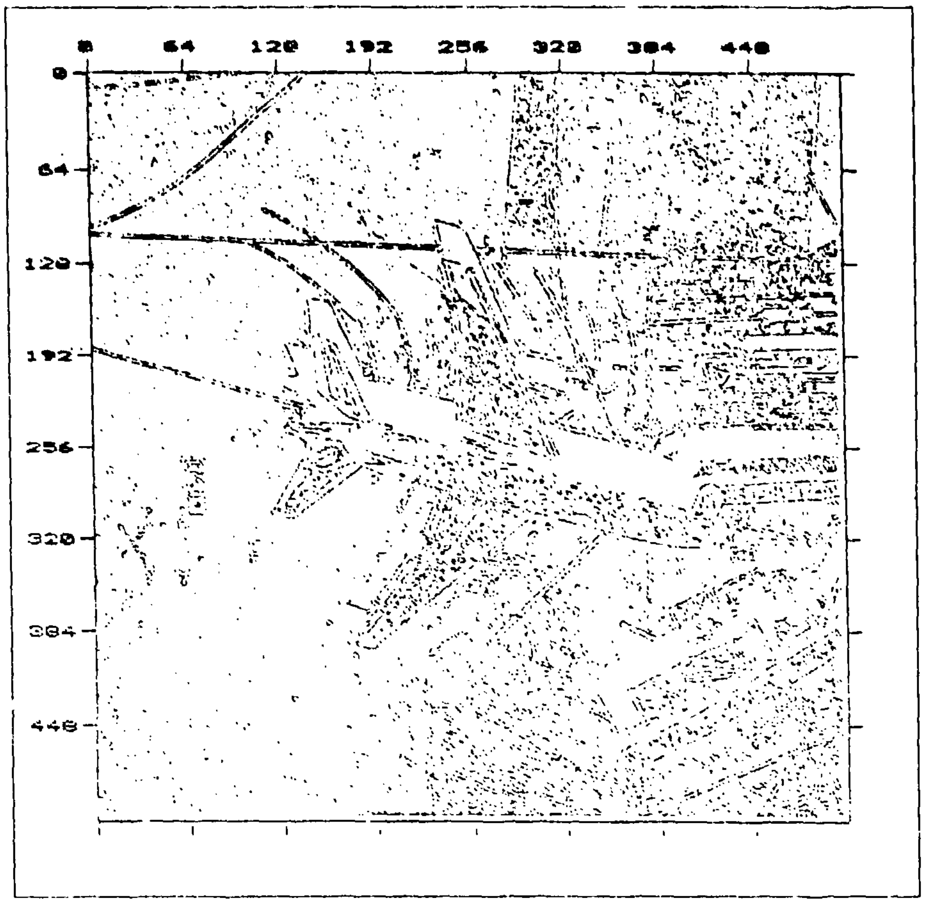
Results

Figure 1 (512 x 512 x 8 bits) depicts an aeroplane on a runway. We have chosen this picture to illustrate our method since results on this image have been presented elsewhere in the literature and it contains many objects of interest for this type of processing.

The result of operating on this image with a 5 x 5 lateral inhibition operator yields the image presented in figure 2.

When the zero-crossings in Figure 2 having a contrast of more than 2 intensity units are found and linked we obtain an outline figure for the laterally inhibited image. Such a figure produced in this way is presented in Figure 3.

By expansion of windows of Figure 3, we can illustrate the precision of our method. These expansions are presented in Figures 4 & 5.



Linked zero-crossing points of laterally inhibited image
Figure 3

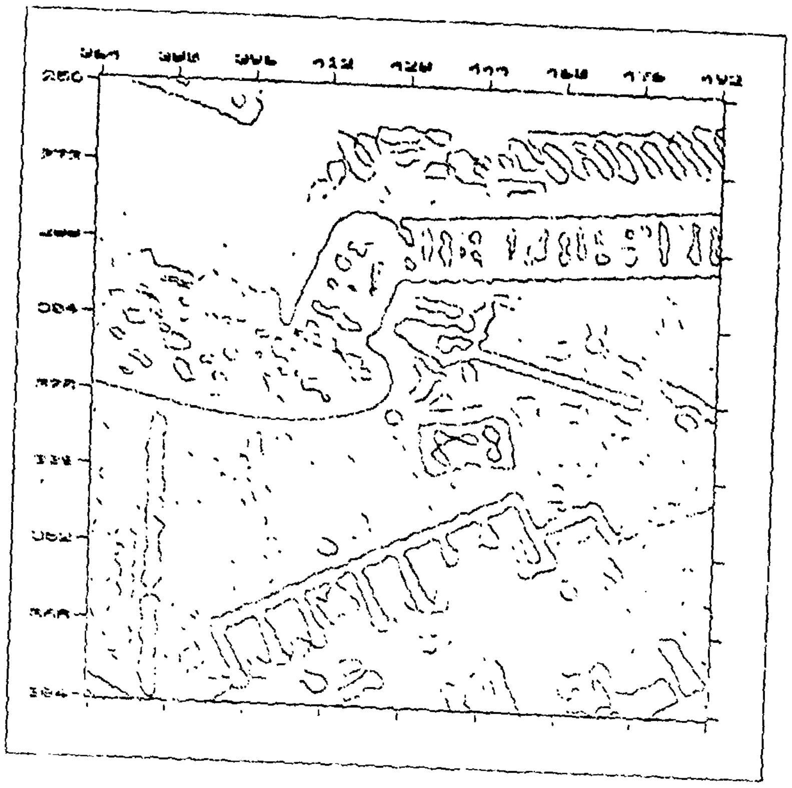
It is interesting to compare the accuracy of the results obtained by the method presented here with known values. Data from Lockheed documents indicates that the wing span for models -1, -100, and -200 is 47.35 m and that the wing span for model -500 is 50.08 m. For all models the tail span is 21.82 m. A convenient measure of the accuracy then is the ratio of the wing span to tail span which for the data above is 0.4619 and 0.4367 respectively. Using expanded images for the wind and tail tips, such as shown in Figure 5, a ratio of 0.4636 was obtained which agrees with the first ratio within 0.4%and with the second ration to within 6.2%. We may therefore conclude that the image was one of the former models. This result would suggest that different planes could except for, perhaps, some very special cases, be easily distinguised and identified using the technique reported here.

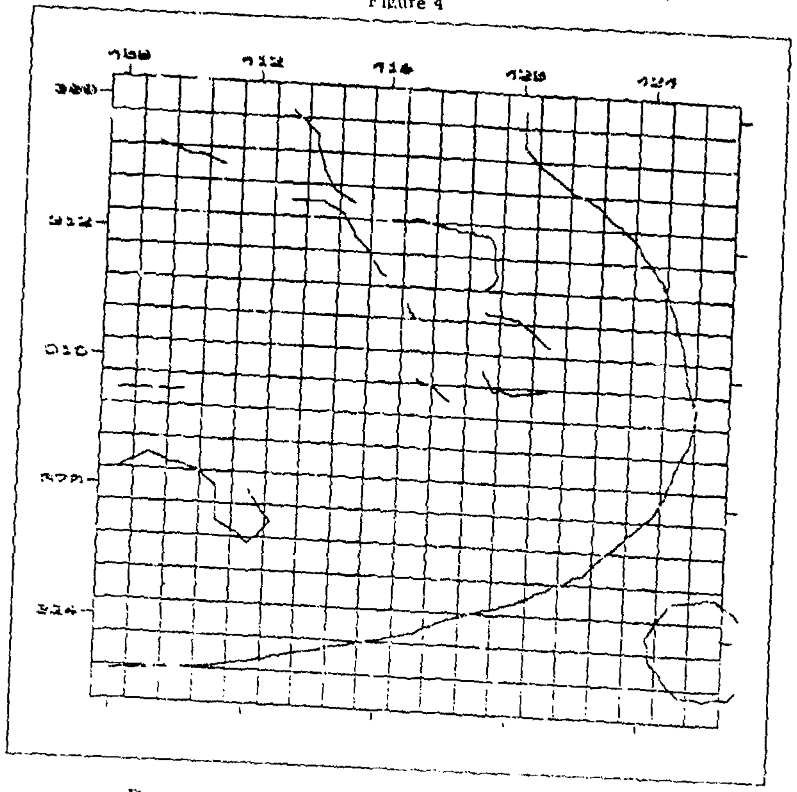
Discussion and Conclusions

As the results show, our processing produces rather improved results over those that have been obtained using other methods. The structure filter was most effective in enhancing the fine structure to be found in the middle right hand side of the image. Improvements in this approach are being pursued.

For a comparison of the results shown here with other methods using the same image as input see Brooks (1979) for results using the Nevatia-Babu technique and Arnold (1978) for the edges obtained using a Heuckel operator.

For comparison with recent results using other images and techniques currently available in the literature, see the results of Rosenfeld (1979) and Tavakoli (1980).


Expansion of a window of Figure 3 (128 x 128)
Figure 4


Expansion of a window of Figure 3 (16 x 16)
Figure 5

We are currently extending this work to provide improved 'real picture' data for some of the other image processing projects being studied in our laboratory such as Arnold's stereo work, ACRONYM (Brooks 1979), and Lowe's work on geometric modeling (these Proceedings).

Acknowledgement This work was performed while P.J. MacVicar-Whelan was on a leave of absence from Grand Valley State Colleges, Michigan. Richard Stadler of Lockheed, Burbank kindly provided over-weekend data on the L1011.

References

1978 **Arnold, R.D.**
"Local Context in Matching Edges for Stereo Vision"
Proc. ARPA Image Understanding Workshop,
Cambridge, Mass. May, pp65-72

1980 **Baker, H.H.**
"Edge Based Stereo Correlation"
Proc. ARPA Image Understanding Workshop,
University of Maryland, 30 Apr pp168-175

1970 **Binford, T.O.**
"The Topologist"
Internal Rept. MIT-AI

1979 **Brooks, R.A.**
"Goal Directed Edge Linking and Ribbon Finding"
Proc. DARPA Image Understanding Workshop,
SRI, Menlo Park, May pp72-78

1980 **Grimson, W.E.L.**
"Computing Shape Using a Theory of Human Stereo Vision"
Department of Mathematics, MIT(thesis), June 1980
MIT AI Lab Memo 565

1970 **Herskovits, A. & Binford, T.O.**
"On Boundary Detection"
MIT AI Memo 183

1973 **Horn, B.K.P.**
"The Binford-Horn Edge Finder"
MIT AI Memo 285

1979 **Marr, D. & Poggio, T.**
"A Computational Theory of Human Stereo Vision"
Proc. R. Soc. Lond. B204, pp301-328

1979 **Marr, D. & Hildreth, E.**
"Theory of Edge Detection"
Proc. R. Soc. Lond. B207, pp187-217
MIT AI Memo No. 518, April 1979

1978 **Nevatia, R. & Babu, K.**
"Linear Feature Extraction"
USC-IPI Rept. 840
Proc. DARPA Image Understanding Workshop,
CMU Pittsburgh, Nov, pp73-78

1979 **Rosenfeld, A.**
"Levels of Represntation in Cultural Feature Extraction"
Proc. DARPA Image Understanding Workshop,
USC, Los Angeles, pp112-127

1980 **Tavakoli, M.**
"Toward the Recognition of Culture Features"
Proc. DARPA Image Understanding Workshop,
University of Maryland, 30 Apr pp33-57

MULTI-RESOLUTION PIXEL LINKING FOR IMAGE SMOOTHING AND SEGMENTATION

Teresa Silberberg
Shmuel Peleg
Azriel Rosenfeld

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

When an image is smoothed using small blocks or neighborhoods, the results may be somewhat unreliable due to the effects of noise on small samples. When larger blocks are used, the samples become more reliable, but they are more likely to be mixed, since a large block will often not be contained in a single region of the image. A compromise approach is to use several block sizes, representing versions of the image at several resolutions, and to carry out the smoothing by means of a cooperative process based on links between blocks of adjacent sizes. These links define "block trees" which segment the image into regions, not necessarily connected, over which smoothing takes place. In this paper, a number of variations on the basic block linking approach are investigated, and some tentative conclusions are drawn regarding preferred methods of initializing the process and of defining the links, yielding improvements over the originally proposed method.

INTRODUCTION

Suppose that an image is composed of a few types of regions each having approximately constant gray level. In principle, the image can be segmented into these regions by gray level thesholding, i.e., by slicing the grayscale into intervals, and classifying each pixel according to the interval in which its gray level lies. However, if the image is noisy, this pixel-by-pixel segmentation process may make many errors, since the noise will cause some of the pixels belonging to one type of region to have gray levels lying in the intervals corresponding to another type. Segmentation could become more reliable if we first smoothed the image to reduce its noisiness.

An image can be smoothed by local averaging, i.e., averaging the gray level of each pixel with the gray levels of a set of its neighbors. However, this process will blur the boundaries between the regions, since a pixel near such a boundary has neighbors lying in both its own region and the adjacent region. If we knew which neighbors belonged to the same regions as the pixel, we could use only these neighbors in the average. In other words, the quality of the smoothing process would be improved if we could first segment the image into the appropriate regions, so that smoothing could be performed within the regions only, not across their borders.

These remarks suggest that it might be preferable to perform smoothing and segmentation concurrently, using some type of cooperative process. An example is the combined smoothing and neighbor linking process defined in [1]. Here weights are assigned to the links between a pixel and its neighbors based on their similarity; the image is smoothed by weighted averaging of each pixel with its highest-weighted neighbors; concurrently, the weights are adjusted as the similarities between neighbors change. The process is iterated, with weighted averaging and weight adjustment alternating. Note that this process does not involve classification of the pixels, but does yield a segmentation of the image into regions based on the connectedness relation defined by the links, if we threshold their weights.

This paper deals with another approach to concurrent smoothing and segmentation based on linking, using versions of the image at different resolutions and defining links between overlapping "pixels" at successive resolutions. In a low-resolution image, the pixels interior to regions have gray levels that are less noisy, since a pixel at low resolution represents an average and is thus less variable. On the other hand, the lower the resolution, the less likely it is that a pixel is contained in a single region; most pixels will overlap two or more regions. The approach considered here, which was first described in [2], takes advantage of both high and low resolutions by using a cooperative process in which the images of successive resolutions interact. A detailed description of this approach will be given in Section 2. A number of variations on the multilevel approach have been investigated, involving changes in the initialization of the process, the method of defining links, and the iteration sequencing; these are described in Section 3.

MULTIRESOLUTION PIXEL LINKING

Let the size of the original image by 2^n by 2^n. To define the reduced-resolution versions of the image, we make use of an exponentially tapering "pyramid" of arrays of sizes 2^{n-1} by 2^{n-1}, 2^{n-2} by 2^{n-2},... 4 by 4, 2 by 2, so that the kth level has size 2^{n-k} by 2^{n-k}. To avoid border effects, all these arrays are regarded as cyclically closed, i.e., the first column is regarded as lying to the right of the last column, and the top row below the bottom row. The elements of each array will be called pixels or nodes. Many different schemes can be defined for constructing such pyramids [3], but in our experiments we used only the simple scheme that will now be described.

We will assign gray levels to the nodes at each level ($k>0$) by taking (weighted) averages of the gray levels of 4-by-4 blocks of nodes at the level below it. The blocks corresponding to adjacent nodes overlap by 50%; this is why the reduction in size from level to level is by a factor of 2, not a factor of 4. For example, suppose node (i,j) at level $k>0$ corresponds to the block of nodes

$$\begin{array}{cccc}
(u,v) & (u+1,v) & (u+2,v) & (u+3,v) \\
(u,v-1) & (u+1,v-1) & (u+2,v-1) & (u+3,v-1) \\
(u,v-2) & (u+1,v-2) & (u+2,v-2) & (u+3,v-2) \\
(u,v-3) & (u+1,v-3) & (u+2,v-3) & (u+3,v-3)
\end{array}$$

at level $k-1$ (where $(u,v) = (2i-1,2j+1)$). Then node $(i+1,j)$ corresponds to the block

$$\begin{array}{cccc}
(u+2,v) & (u+3,v) & (u+4,v) & (u+5,v) \\
(u+2,v-1) & (u+3,v-1) & (u+4,v-1) & (u+5,v-1) \\
(u+2,v-2) & (u+3,v-2) & (u+4,v-2) & (u+5,v-2) \\
(u+2,v-3) & (u+3,v-3) & (u+4,v-3) & (u+5,v-3)
\end{array}$$

where all additions and subtractions are modulo 2^{k-1}. It is easily seen that any node (u,v) below the top level (i.e., $k<n-1$) belongs to four blocks corresponding to nodes on the level above it - in our example, the nodes (i,j), $(i-1,j)$, $(i,j+1)$, and $(i-1,j+1)$. [Note that only for the last of these nodes does (u,v) belong to the center 2-by-2 portion of its block; for the other three, (u,v) is a border point of their blocks.] The level $k-1$ nodes in the block corresponding to a given node at level k will be called its sons, and the level k nodes to whose blocks a given node at level $k-1$ belongs will be called its fathers. Thus every node at level > 0 has 16 sons, and every node at level $< n-1$ has four fathers. Note that since there are only 16 nodes at level $n-2$, each of them is a son of all four nodes at level $n-1$, so that every node in the pyramid is a descendent of every one of these "top" nodes.

The node linking process is as follows: the reduced resolution images are initially defined by unweighted averaging of the gray levels in each block. The gray level of each node is then compared with the levels of its four fathers, and a link is established between the node and its most similar father, i.e., the father whose level is closest to the node's level. After this has been done at every level, we recompute the gray level of each father by averaging only those sons that are linked to it. (If no sons are linked to a father, we give it "gray level" zero.) Based on these new averages, a node's most similar father may have changed, so we next change the links as necessary, then recompute the averages, then change the links again, and so on. Typically, this process stabilizes after a few iterations.

To see how the process works, let us define the base of a node as the set of pixels that are linked (through as many intermediate stages as necessary) to that node. Thus initially the base of every node is a square block of pixels. If the base of a node initially lies mostly inside a region, the node is most likely to become linked to nodes on the level below that also lie (mostly) in that region; thus its recomputed average will become closer to the region average. As the process is iterated, nodes at relatively high levels acquire values that approach the average values of regions, even though they are too large to fit into a region. Slight initial biases in the node averages at high levels will result in high-level nodes being driven toward values that correspond closely with the averages of regions or sets of similar regions in the image. For further discussion of the process, see [2].

Supposed that there are not more than four types of regions in the image. For each type, there should be at least one node at the top level of the pyramid whose average converges to the average gray level of the regions of that type. This node will be linked to nodes which are linked to nodes ... which are linked to the pixels belonging to these regions. In other words, this node becomes the root of a tree whose leaves are the pixels that lie in regions of the given type. If there are fewer than four types of regions, there may be two such trees corresponding to the same region type, representing different subsets of the pixels in these regions. If we know how many region types there are supposed to be, we can suppress some of the nodes at the top level (i.e., forbid anyone to link to them), keeping only as many top-level nodes as there are types. [Alternatively, we can "merge" some of the top-level nodes together, averaging together their values and using this average as the value for each of them.] In this way, we can insure that the number of trees (having distinct values) is the same as the desired number of region types.

In summary, the iterative linking and averaging process is defined as follows:

a) Initialize the node values by simple block averaging of each node's 16 sons

b) Link each node to that one of its four fathers whose value is closest to its own

c) Recompute the node values by averaging the values of only those sons that are linked to the node

d) Change the links in accordance with these new values

e) Repeat steps (c-d) as many times as desired. Typically, there is little change after the first few iterations, and there is no change at all after 10 or 15 iterations.

At any stage of this process, the links define a set of (up to four) trees rooted at the top level of the pyramid, and we associate with each pixel the value at the root of its tree. Thus the process smooths the image to an extreme degree, giving each pixel its tree average as a smoothed gray level. At the same time, it segments the image into (up to four) subsets, where each subset consists of the pixels which are the leaves of one of the trees.

The smoothing and segmentation accomplished by this process can be compared with those achieved by the pixel linking process of [1]. In [1] the links are all at the pixel level, and the smoothing is local. Even if the link strengths all converged to values of 1 (within a region) and 0 (between regions), many iterations would be required to obtain the global average of each region at each pixel of the region, since it takes O (region diameter) iterations for information to propagate across the region. In the process described here, on the other hand, the links are between levels, and information can propagate "across" a region in O (log region diameter) iterations, since nodes comparable in size to the region are only (log region diameter) levels above the pixel level. Moreover, in our process, smoothing can take place even over sets of non-connected regions of the same type, whereas the process of [1] can smooth only within a connected region.

The concept of linking each node with its most similar father may be compared with the smoothing processes described in [4-6], where a set of neighborhoods lying on various sides of a pixel are examined, and the pixel's value is replaced by the average of the least variable of these neighborhoods (since this neighborhood presumably lies almost entirely within the pixel's region). Using the most similar neighborhood (i.e., the one whose average is closest to the pixel's gray level), rather than the least variable neighborhood, would probably work well too; but we could not use the least variable neighborhood in our scheme. In any event, the methods of [4-6] use neighborhoods of only a single size, which limits the speed with which the smoothing can propagate, as discussed in the previous paragraph.

VARIATIONS

In the experiments described in this section, several variations on the basic pyramid linking process were tried. These variations were concerned with how to initialize the node values; how to choose the father to which a node is linked, and in particular, what to do in case of ties; and how the iteration process is sequenced. In the following paragraphs we describe the variations, and then show the results obtained by using combinations of these variations on a standard set of images (which were also used in [2]): an infrared image of a tank, a portion of a blood smear, and a portion of a chromosome spread. These images are shown in Figure 1 (a-c). All results are shown for a stage at which the iteration process has stabilized; this is usually after about 10 iterations.

a) Initialization. In the method used in [2], the value of each node was initialized by averaging the values of all 16 of its sons. An alternative which (as we shall see) seems to give better results is to initialize by averaging the values for only four of the sons, namely those whose positions in the image are closest to that of the node. (The position of a node is understood to be at the center of its block.) Note that in this alternative scheme, the initial averages are all nonoverlapping.

b) Father selection. In the method of [2] each node is linked to the father closest in value to the node. A more general idea is to take into account both closeness in value and closeness in position. We can compute link merits based on a formula such as $\Delta(D+s)$, choosing the father for which $\Delta(D+s)$ is smallest, where Δ is the difference in value, D is the Euclidean distance between positions, and s is a parameter which is used to vary the effect of the D contribution (for large s, differences in D have little effect).

b') Ties. If two fathers have the same link merits, we resolve the tie based on any arbitrary ordering of the fathers, e.g., NW, NE, SE, SW. The choice of this ordering should not significantly affect the results.

c) Sequencing. In [2], links are determined for all levels; then averages are recomputed for all levels; and this process is repeated. An alternative is to iterate level by level: as soon as the links from the nodes at level k are redefined, the averages at level k+1 are recomputed, and the links from level k+1 are then redefined based on these new averages.

d) Top level nodes. The number ($\leq$ 4) of nodes used at the top level should be the same as the desired number of region types - 2 for the tank and chromosomes, 3 for the blood-cells. We can insure that only two or three nodes at the top level are used by initializing the values of the remaining node(s) to a very high number, thus insuring that no nodes will ever link to them. As a refinement, we can fix the top-level nodes that we do use to have values that represent estimates of the expected region averages; we will show some examples using this variation. We will also show examples of results obtained when we use all four nodes at the top level, even though the desired number of region types is less than four. As we shall see, the process then tends to create somewhat artificial discriminations within the regions.

We first show the results obtained when we use the desired number of nodes at the top level, but do not attempt to set the values of those nodes to the expected region averages. Figure 2 (top two rows) shows these results for the four combinations of initialization and sequencing schemes. We see that in the chromosome case (Figure 2c), four-son initialization gives better results; when 16-son initialization is used, some of the small chromosomes are lost, probably because too much of the background is initially averaged with them, so that they link to a top-level node whose value converges to the background value rather than to the chromosome value. The initialization scheme has little effect on the results for the other two images, and the iteration sequencing scheme has little effect on any of the images. The order used for tie-breaking also has little effect, as we see from the bottom left pictures in Figure 2 (which use the same initialization and sequencing schemes as the top left pictures). Finally, the bottom right pictures in Figure 2 show what happens when we give some weight to Euclidean distance (s=5) in choosing the links (otherwise, same as top left); note that this too improves the results in the chromosome case, and has little effect in the other two cases. It seems from these results that four-son initialization is preferable to 16-son initialization, and that it may also be preferable to give some weight to Euclidean distance in choosing links; but the other variations make little difference. The exact shapes of the tank and cell nucleus are somewhat sensitive to variations because the correct links for blocks near the borders of these regions will be somewhat ambiguous, due to the noisiness or texturedness of the regions.

Figure 3 shows analogous results when the top-level nodes are given estimates of the average region gray levels as fixed values. Again, the variations make little difference for the tank and cell images, but they are significant for the chromosome image. The loss of the small chromosome has now become dependent on the iteration sequence and even on the tie-breaking order (!); and when we use the four-son initialization method, a large chromosome is lost (!). Apparently, attempting to fix the values of the top-level nodes as equal to the estimated region averages can actually degrade the performance of the pyramid linking process.

Figure 4 gives analogous results when all four top-level nodes are used, so that the process tries to find four region types in each image.* The resulting artifacts are especially apparent for the chromosome image, where the background gets segmented into three subregions that differ appreciably in average gray level. Here again, when we use 16-son initialization, the small chromosomes become part of the background, but this does not happen when four-son initialization is used, nor when weight is given to Euclidean distance in choosing links. The other variations have little effect, and all of the effects are minor for the cell and tank images (the tank region splits up into "noisy" subregions in various ways, but does not get badly confused with the background). Thus these results support the conclusions derived from Figure 2.

*If the input image contains fewer than four gray levels (e.g., if we threshold the chromosome or tank image into two levels or the cell image into three), the process does not create additional values, even though all four top-level nodes are used.

When the process is applied to a perfectly regular input pattern such as a checkerboard, it breaks down and fails to segment the pattern into two region types, unless ties are broken <u>randomly</u>. Figure 5 shows results analogous to those in Figure 4 (left column corresponds to Fig. 4 top left, and right column to bottom right), but using random tie-breaking: the results are quite similar.

The smoothing effect of the process as we follow the links from level to level can be assessed by constructing histograms corresponding to each level's view of the image. Suppose that, for a given k, we give each pixel a gray level equal to the value of the node at level k to which it is linked. When we do this for k=0, 1, 2, ..., we obtain a sequence of successively smoother and simpler images, whose histograms become successively more spiky, until finally, the histogram obtained from the top level consists of (at most) four spikes. Such histograms for the three images, after one iteration of the linking and reaveraging process, are shown in Figure 6 for levels 0, 1, 2, 3, 4. (16-son initialization was used, and links were chosen based on value similarity only). If we did not want to rely on the iterative process to converge to a good segmentation, we could still consider using a single iteration of the process to improve the separation of the histogram peaks, so that segmentation by thresholding based on the histogram would be easier.

CONCLUDING REMARKS

This paper has investigated a number of variations on the basic pyramid linking process. The results suggest the following tentative conclusions:

a) It appears to be preferable to use schemes in which some weight is given to the relative positions of nodes, both in initializing their values and in choosing links, especially in cases involving regions that consist of many small connected components. Apparently, when we take relative positions into account, we have a better chance of preserving the integrity of small regions.

b) It is desirable to specify the desired number of region types (i.e., pixel classes), by allowing only that number of nodes at the top level to be "active." Otherwise, the process tends to split some of the classes artificially. On the other hand, using estimates of the average gray levels of the classes to fix the values of the nodes at the top level may degrade the results, perhaps because it introduces premature biases that are not compatible with the early stages of the linking pattern.

c) The process is relatively insensitive to the sequencing of the iterations and to the node ordering used for breaking ties.

The experiments reported in this paper have led to a better understanding of the pyramid linking concept. The conclusions will serve as guidelines in the design of linking processes based on pixel properties other than (average) gray level, for application to the smoothing and segmentation of multispectral or textured images.

REFERENCES

1. J. O. Eklundh and A. Rosenfeld, Image smoothing based on neighbor linking, IEEET-PAMI 3, 1981, in press.

2. P. Burt, T. H. Hong, and A. Rosenfeld, Segmentation and estimation of image region properties through cooperative hierarchical computation, TR-927, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, August 1980.

3. P. J. Burt, Fast hierarchical correlations with Gaussian-like kernels, TR-860, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1980.

4. F. Tomita and S. Tsuji, Extraction of multiple regions by smoothing in selected neighborhoods, IEEET-SMC 7, 1977, 107-109.

5. M. Nagao and T. Matsuyama, Edge preserving smoothing, Computer Graphics Image Processsing 9, 1979, 394-407.

6. R. M. Haralick and L. Watson, A facet model for image data, Proc. PRIP 79, 489-497.

(a) (b) (c)

Figure 1. The three images used in the experiments: a) tank, b) blood cells, c) chromosomes.

Figure 2. Effects of varying the initialization, sequencing, tie-breaking rule, and linking criterion (see text).

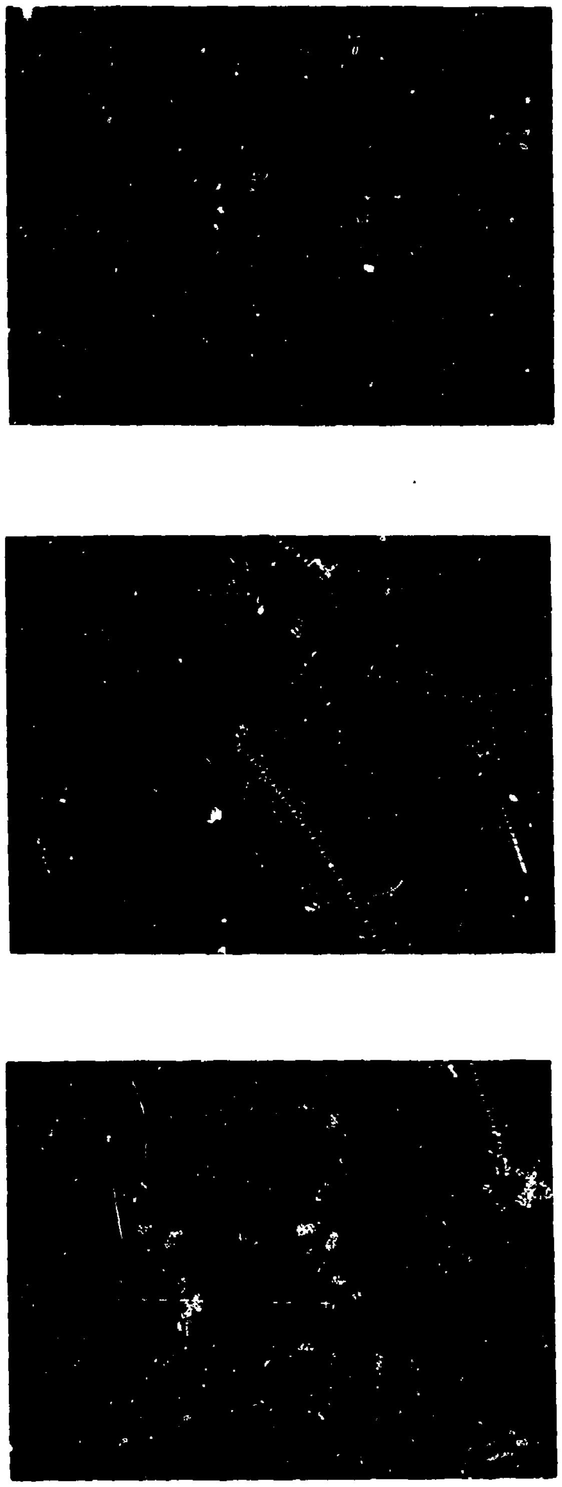


Figure 3. Analogous to Figure 2, but initializing the top-level nodes with estimates of the region averages.

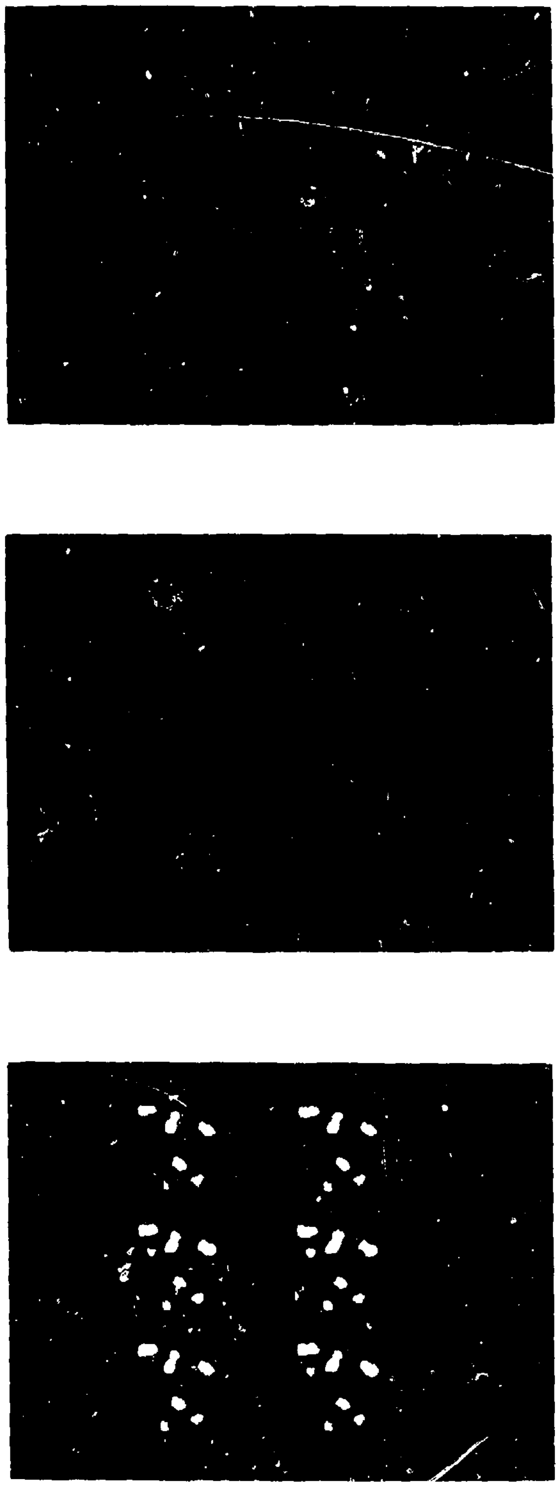
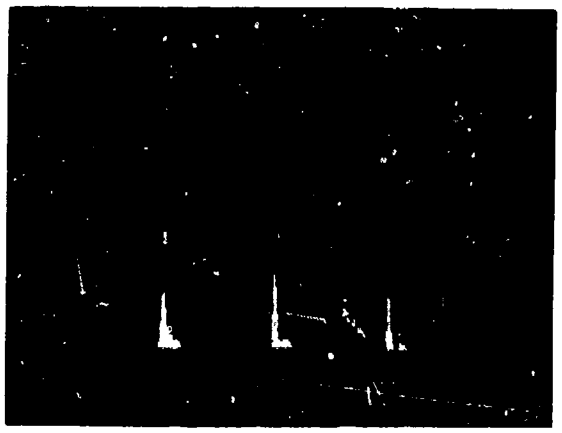
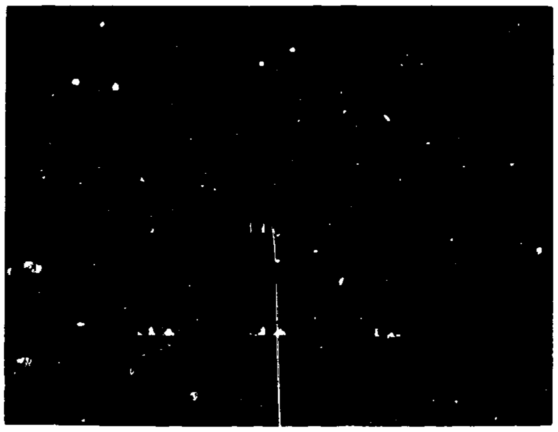


Figure 4. Analogous to Figure 2, but using all four nodes at the top level.

Figure 5. Analogous to the top left and bottom right pictures in Figure 2, but breaking ties randomly.





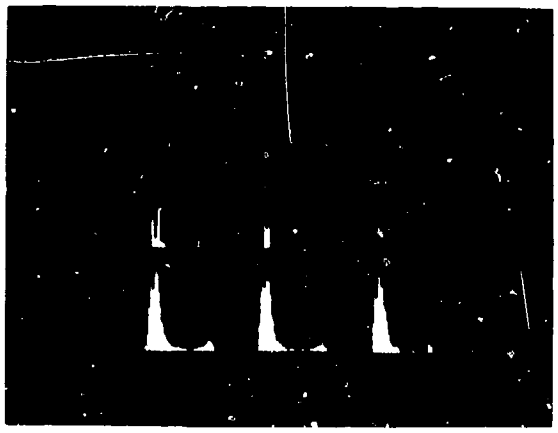


Figure 6. Histograms obtained, after one iteration of linking and re-averaging, when the node values at a given level are assigned to the pixels having those nodes as ancestors, for levels 3 4 .

0 1 2

The s[illegible] t of the Defense Advanced Research Project [illegible]ncy and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76-C-0138 (DARPA order 3206) is gratefully acknowledged. The authors also acknowledge the help of Tsai-Hong Hong in investigating the effects of the linking process on the histogram of the image.

THE INTERPRETATION OF GEOMETRIC STRUCTURE FROM IMAGE BOUNDARIES

David G. Lowe and Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

General constraints on the interpretation of image boundaries are described and implemented. We illustrate the use of these constraints to carry out geometric interpretation of images up to the volumetric level. A general coincidence assumption is used to derive suggestive but incomplete interpretations for local features. A reasoning system is described which can use these suggestive hypotheses to derive consistent global interpretations, while maintaining the ability to remove the implications of hypotheses which are disproved in the face of further evidence. An important aspect of interpretation is the classification of image boundaries (intensity discontinuities) into those caused by geometric, reflectance, or illumination discontinuities. These interact with other hypotheses regarding occlusion by solid objects, the direction of illumination, aspects of object geometry, and the production of illumination discontinuities by geometric discontinuities. Although only a subset of the constraints and system design features have been implemented to date, we demonstrate the successful interpretation of some simulated image boundaries up to the volumetric level, including the construction of a three-space model.

Introduction

This paper describes work in progress on the derivation and use of general constraints on the interpretation of image curves (also known as intensity discontinuities or lines). These constraints are derived from general assumptions regarding illumination, object geometry, and the imaging process, and are carried out to the level of three-dimensional volume interpretations. This work is being done in the context of the ACRONYM vision system [2], and is expected to interface closely with the higher levels of ACRONYM, which make interpretations from generic object models. The current implementation of these constraints in an independent computer program is described and demonstrated.

Much of the previous work on the interpretation of image lines has concentrated on the constraints imposed on boundary junctions by certain classes of geometric objects [3, 5, 6, 7]. We believe that there are more general constraints on the formation of image boundaries, and that the use of these constraints is of great importance for the interpretation of real data and general classes of images. Most previous attempts at boundary interpretation have used connectivity as the main source of information, whereas we have placed at least as much emphasis on shape, size and location. We also make some use of the intensity contrast across boundaries. Our implementation of these constraints is designed so that no single constraint is taken to be absolute, and the system can therefore tolerate some errors and incompleteness in its initial data.

It is important to realize that there is seldom a unique interpretation for any local image property. All of the constraints which we use are in the form of coincidence assumptions, in which some property suggests a particular interpretation unless some set of coincidences (or errors in the data) have occurred. This means that our reasoning can not be strictly deductive and monotonic (as was, for example, the constraint system used by Waltz [7]), since we must make and use hypotheses while retaining the option of having further information prove them false. Therefore, an important aspect of this work on image interpretation has been the development of a reasoning system which can operate cleanly and efficiently with these incomplete constraints. On the other hand, the search space for this problem is not large, since most of the constraints have a small branching factor and there is typically much redundant information available in support of an interpretation.

We categorize image intensity discontinuities into three distinct classes: those caused by discontinuities in the geometry of an object (edges), in the reflectance of an object (markings), and in the illumination (shadows). The constraints on each of these categories are quite different, and much of the discussion in this paper will deal with the recognition and separate implications of these different classes of image curves.

Reasoning on the basis of coincidence assumptions

As mentioned above, the use of incomplete constraints forces us to use a non-monotonic reasoning system, in which further evidence can disprove a previously held hypothesis. Therefore, the method of deduction we have chosen has been to form an explicit symbolic representation of each hypothesis, and to maintain a record of its support and implications so that it can be reevaluated and undone if new contextual evidence indicates that the original hypothesis was false. Each hypothesis has pointers to all the hypotheses and types of evidence which have given support to it, as well as pointers to the hypotheses which it supports.

Figure 1 shows the data structure representation for a typical hypothesis (in this example the hypothesis is that a particular curve represents a geometric edge causing occlusion on a specific side). The hypothesis in this example is supported by evidence from several other hypotheses (represented in the EVIDENCE slot) and makes use of three basic constraints (to be described in detail in the following section): that the geometric discontinuity corresponds to an intensity discontinuity in the image, that the occlusion is suggested by the termination of another geometric edge at this geometric edge, and that this edge casts a certain shadow. The SUGGESTS slot of the data structure points to all hypotheses which are based at least in part on this one.

When conflicting interpretations are suggested for some part of the image, the evidence for each interpretation is evaluated to see if one interpretation is strong enough to exclude the other. If there is insufficient information to make this decision, then both hypotheses are pursued until there is. When a previously held hypothesis is thought to be false, its implications are undone and any further results propagated. This non-committal type of reasoning appears to be flexible and easy to use. The major difficulty is that this requires that special code be written to evaluate and resolve each type of conflict that could occur, but we are looking at other possible resolution schemes. The explicit maintainance of constraints is much better than the use of backtracking for testing hypotheses, since it is not limited to the last-in-first-out order of testing typically enforced by a backtracking stack, and any results apply to the entire problem space.

The coincidence assumptions could sometimes be used to measure the likelihood that a given coincidence will occur, and therefore be used to compute a probabilistic degree of confidence in a hypothesis. However, these probabilistic measures can be extremely context-dependent and may vary widely from image to image in ways that can not be known before the image is interpreted (ie., the probabilities cannot reasonably be assumed to be independent). The combination of probability values may be of importance in some cases for speeding up the analysis, by telling us what to examine first, but we have made no attempt to use them in our current system. When some consistent overall interpretation has been found for an image, there is usually much redundant information available in support of the interpretation and there is no need to choose between alternatives on the basis of explicit probabilities. Our set of hypotheses resemble the "blackboard" form of hypothesis representation [4] in many ways; however, we attempt to resolve conflicts by pursuing each possibility independently rather than by combining probability estimates, and we maintain the explicit history of each hypothesis in symbolic form.

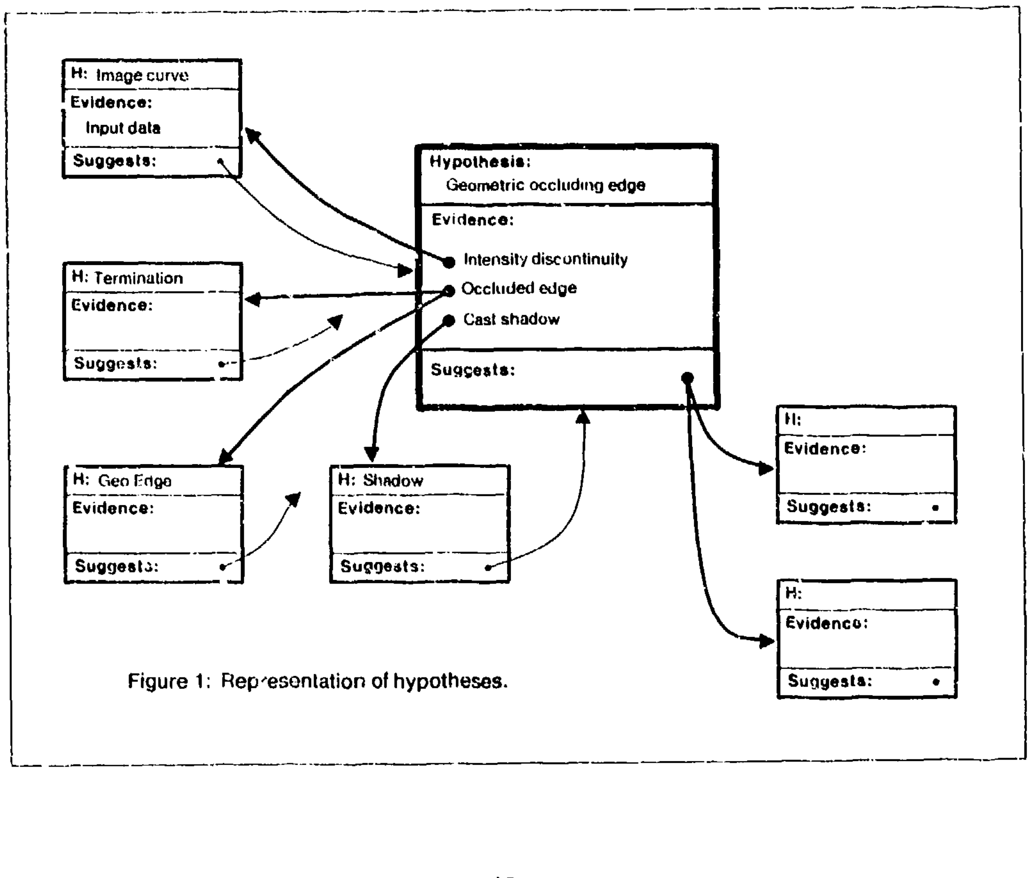


Figure 1: Representation of hypotheses.

Constraints on the interpretation of image curves

This section describes a series of general constraints on the three-space interpretation of image lines. In each case we attempt to describe the specific coincidences which would lead to alternate interpretations. Many of these constraints are developed in more detail in [1].

In a certain trivial sense, any of the image features described in the following constraints could be the result entirely of reflectance discontinuities rather than geometric or illumination discontinuities, since the image could be a picture of a picture. However, interpretations which are consistent with solid geometry and illumination are unlikely to arise from surface markings, except when those markings have been specifically designed to correspond to images of geometric objects. Therefore, we treat surface markings which have consistent geometric interpretations as coincidences which are hypothesized only when given other evidence.

The "curves" described in the following constraints are assumed to separate regions of different intensities. Regions which are too narrow to have a measurable width ("wires" or thin lines marked on a surface), although often represented as ordinary lines in a line drawing, are locally different from other types of curves in a digitized image and are localized by different criteria [see 1]. They should be treated as thin occluding regions, even though the precise width of the regions may not be accurately known. If we wish to interpret line drawings in which there is no distinction locally between thin regions and boundaries separating regions of different intensities, then the following constraints can be easily modified to incorporate this possibility.

1) Tangent discontinuities in curves. Breaks in an image curve (also known as tangent discontinuities or corners) are important for the same reason that intensity discontinuities are important in an image: they are features which can be sharply localized, and can therefore lead to more restrictive constraints than more diffuse features. Two-dimensional breaks in an image curve are closely related to breaks in three-dimensional space curves: in fact, a smooth space curve cannot have a break in its image. Likewise, a break in a space curve will result in a break in its image, unless there is a coincidence in which the observer is coplanar with the two tangents. The situation is the same for a cast shadow: there will be a break in the shadow cast by a geometric edge which contains a break, unless the source of illumination is coincidentally coplanar with the two tangents of the break.

2) Straight lines. An image line which is straight must be the image of a straight space curve, unless the curve is planar and the observer is coincidentally aligned with the plane of curvature. Likewise, a straight shadow curve in the image must have been cast by a straight geometric boundary onto a planar surface, unless the source of illumination is coincidentally in the plane of curvature of the geometric boundary or the observer is in the plane of curvature of the shadow curve.

3) Termination at a continuous curve. When an image curve terminates at a continuous curve (a T junction), the terminating curve cannot be closer to the observer than the continuous curve; otherwise, it would be a coincidence that the termination happened to occur on the other line. The T junction could be the result of three different occurrences: occlusion of the terminating edge by a geometric boundary, the termination of a surface marking at the visible edge of a geometric object, or a specific set of surface markings. Therefore, if we know that the terminating curve is a geometric boundary, then we can infer that the continuous curve is also a geometric boundary and we know its direction of occlusion. If we know that the continuous curve is a geometric boundary occluding on the side of the terminating curve, then we can infer that the terminating curve must be a surface marking. If we know that the terminating curve is a shadow, then we can infer that the continuous curve is a non-concave geometric boundary (if it was concave we would see a continuation of the shadow from the same vertex).

4) Crossing of continuous curves. When two continuous curves cross one another (an X junction), this is an indicator of either an illumination discontinuity, transparency, or an unusual combination of surface markings, since the curve closest to the observer does not occlude either side of the other (note that we are assuming other local evidence for wires, as mentioned above). If one of the curves is an illumination discontinuity (shadow boundary) this implies that the other curve lies on the same surface and is not a geometric boundary, since otherwise a break would occur (as will be discussed further below).

5) Contrast across shadow edges. The contrast across a shadow edge is equal to the ratio of direct to indirect light, which remains fairly constant across an image for a distant light source. However, the situation is complicated by the presence of reflections from nearby objects, which can considerably increase the amount of indirect illumination. A stronger constraint is that the contrast ratio along the length of a shadow will change only smoothly and independently from the surface on which it falls, so that as the shadow curve crosses different reflectance boundaries (or even crosses illumination boundaries cast by secondary sources of illumination) there will be the same contrast ratio on both sides of the boundary. This is a valuable constraint for identifying shadow lines. In the case of the X junction mentioned above in (4), the illumination discontinuity will have the same contrast ratio on both sides of the junction, and can therefore be distinguished from the other curve at the X junction, barring coincidence. In color imagery, the ratios of illumination at each wavelength across a shadow boundary should be fairly constant and vary only smoothly along the length of the shadow, and could therefore provide an even stronger constraint (eg. the dark side of a shadow boundary will be bluer than the sunlit side on a clear day).

6) Shadow breaks caused by geometric breaks. As mentioned in (1), barring coincidence, breaks (tangent discontinuities) in geometric edges are observable as breaks in image lines, and geometric edges with breaks cast shadows with breaks. Therefore, given a known direction of illumination, the geometric break causing any particular shadow break is constrained to lie in a precise direction. This constraint is so strong that the mere existence of a break in the given direction is support for the hypothesis of a shadow. If the direction of illumination is unknown, then almost parallel sets of matching breaks between hypothesized shadows and other image lines provide evidence for the hypothesis of the direction of illumination (it can usually be assumed that the source of illumination is moderately distant from at least some parts of an image).

In perspective imagery there is an *illumination convergence point* in the image through which the images of all illumination rays from a point source pass (this is true even for nearby point sources). The location of this point can be determined from any set of two or more matches between shadow and geometric breaks. If the point source is in front of the camera lens plane, then the convergence point is of course the location of the image of the point source. If the light source is behind the camera lens plane, then the illumination convergence point is located at the point of projection of the light source onto the film plane through the projective center of the camera, and the illumination streams towards this point rather than away from it. If the point source is exactly in the lens plane of the camera, then the perspective effect compensates for divergence from the light source to make the illumination convergence point infinitely far off, so all images of illumination rays are exactly parallel (these insights are due to Sid Liebes).

It should be noted that breaks in shadow curves can also be caused by the intersection of shadows cast by different objects, so it cannot be assumed that there is a geometric break corresponding to every shadow break. The constraints given here could be extended to cover discontinuities in curvature and other shape properties as well as discontinuities in tangents, although these are more difficult to implement computationally.

7) Casting of shadow curves by geometric edges. Given one or more matches between shadow breaks and geometric breaks it is easy to determine which geometric edg[illegible] are casting which shadow edges. An important constra[illegible] that each point on the shadow edge must correspon[illegible] [illegible] some point on the geometric edge in the direction toward or away from the illumination vanishing point, although all parts of the geometric edge may not be observable (say, if it is occluded by another object). Once a match has been hypothesized between a shadow edge and a geometric edge, many important inferences follow: the casting curve is known to be a geometric edge or limb with the direction of occlusion depending on the direction of contrast across the shadow edge. If the geometric edge is straight then any curvature in the shadow curve is due to curvature in the surface on which it is cast. Likewise, if the shadow surface is known to be planar, the curvature of the shadow can be used to calculate the curvature of the casting edge. The image separation of the casting curve from the shadow can be used to calculate their relative range from the camera, as will be described later. Shadows essentially provide a second projection of the object geometry, in addition to the original image, and can therefore be used in much the same way as stereo information.

8) Junctions of two or more discontinuous curves. When two or more curves terminate at the same junction, it would be a co[illegible]cidence if the observer were aligned so that separated vertices in space landed at the same point in the image. Therefore, for L, Y, K, or higher-order junctions, it is reasonable to hypothesize that coincidence in the image implies coincidence in space. In other words, knowing constraints on the 3-space location of the endpoint of any of the terminating curves gives us the same constraints on the 3-space locations of the other endpoints at the vertex. When a shadow curve terminates at a Y or higher-order junction, we can assume that one of the other terminating curves is a geometric edge casting this shadow onto a surface passing through the junction (otherwise it would be a coincidence that the shadow happened to pass through the junction).

9) Propagation of direction of occlusion. When the direction of occlusion is known for a geometric boundary (ie., we know which of the surfaces on either side the edge belongs to), then any unambiguous continuations of the geometric boundary will have the same direction of occlusion. As long as the curve is continuous we assume continuity in space, and by (7) we also assume continuation at L junctions.

10) Surface continuity. We assume smoothness and continuity of surfaces when there is no intensity discontinuity in the image. If an intensity discontinuity on a surface is due to a geometric discontinuity of the surface, then we would expect to see a discontinuity in the geometric boundary of the surface where it intersected this intensity discontinuity (unless the observer is in the plane of the two tangents at the boundary).

Knowing the direction of occlusion (from 3,7,9, etc.) allows us to form surface descriptions by looking at the regions bounded by geometric boundaries. In many cases it is possible to form surface descriptions by merely following the continuations of geometric boundaries around the extent of the surface, until they are possibly occluded by another surface. Another technique is to work away from the occluding side of a geometric boundary, ignoring illumination and reflectance discontinuities, until either an opposing geometric boundary is found or an occluding geometric object is found Note that this is different than the usual region segmentation of images, since we are doing it in the geometric domain. These can be difficult constraints to implement computationally, since they deal with properties of entire surfaces rather than just a curve. Our current implementation handles only some of the more common cases of surface description.

11) Alignment of image curves. When two straight lines are aligned in an image (even though they may be separated by a gap of a considerable distance), they must also be aligned in space, or else the lines must be parallel and the observer must be coincidentally in the plane of the two lines. This constraint allows us, for example, to hypothesize continuity of line segments on both sides of an occluding object. This constraint can be extended to deal with the alignment of circular arcs, elliptical curves, repetitive textures, symmetries, and any shapes which are predicted from hypotheses and knowledge of the imaged object. This constraint can also be used to bridge gaps in curves due to errors in the curve detection process or insufficient contrast across some part of a boundary. This is an important area for further research.

12) Prediction of illumination boundaries. If we have formed hypotheses of the geometry of an object, the surrounding surfaces, and the direction of illumination, then we can predict the locations of illumination discontinuities in the image, and thereby produce new hypotheses for image lines at these locations, as well as confirm or contradict our original hypotheses. Prediction of this sort can also be used to check consistency of surface occlusion.

Since we are attempting to derive three-dimensional structure from image clues, the constraints above all deal with image features which are *quasi-invariant* with respect to viewpoint (eg., a curve break in three-space produces an image curve with a break over a wide range of viewing conditions, even though the angle of the break is not invariant). The higher levels of ACRONYM produce predictions of quasi-invariants from specific object models, and these should interface well with the more general constraints described above.

The list given above is far from complete. We are particularly interested in developing new constraints for infering the cross sections and volume descriptions for the geometric objects in an ima. Many other image features (such as parallelism) are only slightly less invariant with respect to viewpoint, and should also be included in a general system.

An Implementation

We have written a preliminary version of a computer program which implements the hypothesis formation system and many of the constraints described above. This program has been tested on simulated image curve data derived by hand from a real image, with the encouraging results described below. We are in the process of implementing more constraints, and hope to soon test the program on data derived automatically from images by curve detection programs being written here at Stanford. The program has been implemented in MACLISP on a DEC KL-10, using the record package and ACRONYM environment created by Rod Brooks [2]. All of the constraints we currently use are computationally inexpensive, and the hypothesis formation for the example given on the following pages took less than 2 seconds of computer time.

The initial curve data is translated into a set of "curve hypotheses" in order to initialize the hypothesis generation process. Each curve is represented as a series of points with the tangent of the curve given at each point, and cubic splines are assumed as the method of interpolating for intermediate points. All curves are indexed in a grid array under all grid squares through which they pass, and it is therefore economical to search image neighborhoods for curve features. During input, each curve termination is linked to other curves which terminate or pass through the same spot. When using real data it may be useful to hypothesize extensions of curves which do not terminate at a vertex.

A weakness of the current system is its control structure. Currently, most of the hypothesis generation proceeds in a fairly fixed order, with the most reliable types of evidence being examined first. However, we are in the process of creating a more flexible control structure using a generalized agenda for ordering constraint propagation. The conflict resolution mechanism also requires further work before it can be applied to all the different types of conflicts which could occur.

In spite of the incomplete state of the current system, it is able to carry out detailed interpretations from simulated curve data as shown in Figures 2 through 10. Figure 2 shows the original picture taken over San Francisco airport from which the curve data in Figure 3 was derived by hand. This data also specifies the approximate contrast across edges. In deriving this curve data we referred to data derived automatically by curve detection programs, and we believe data of at least this quality will soon be available. We have also input the location of the sun, although we hope to soon be able to derive the illumination direction(s) automatically from the image. Figure 4 contains a circle over the location of each hypothesis corresponding to the unambiguous continuation of a curve boundary (constraint 8 of the previous section). It also shows the places at which the input routines segmented the curve data. Figure 5 shows the hypotheses regarding curve terminations at a continuous curve (constraint 3).

Figure 6 shows a dotted line at the match between each hypothesized shadow discontinuity and its corresponding edge discontinuity in the sun direction. This makes a small amount of use of the input data regarding the contrast across curves (constraint 5), but the major support for a shadow hypothesis is the existence of one of these shadow matches (constraint 6). We expect to be able to use tighter tolerances when using automatically generated data than were used with this simulated data, with correspondingly better results. Figure 7 shows curves which have been hypothesized to be shadows as dotted lines, and also the hypothesized connections between shadow lines and the geometric edges which cast them (constraint 7). These hypotheses were formed from the evidence of Figure 6 in combination with other projective constraints mentioned in the previous section and hypotheses regarding which edges constituted geometric boundaries. The success of this technique can be shown in the closeup view of the tail section in Figure 8, in which the complicated outline of the shadow is correctly interpreted as the intersection of shadows from several different geometric edges. Further hypotheses are also formed regarding the planarity of regions, and their direction of occlusion.

From the results of this interpretive process, it is possible to construct three-dimensional models of the scene. The shadow to curve matches provide information on the relative distance of each geometric edge and shadow surface from the camera. Even if the sun position is not known, all

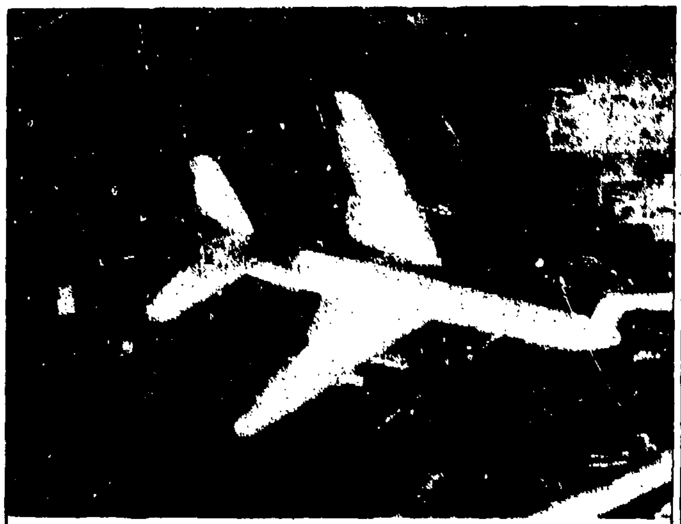


Fig 2 Original digitised image.

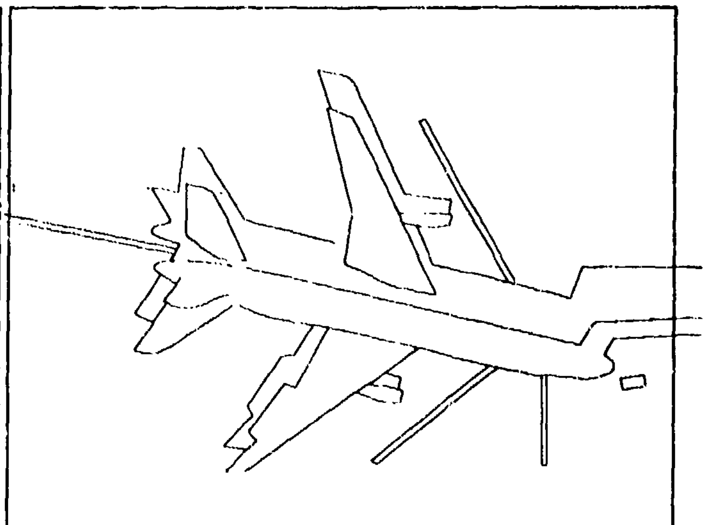


Fig 3 Hand derived curve data used as input.

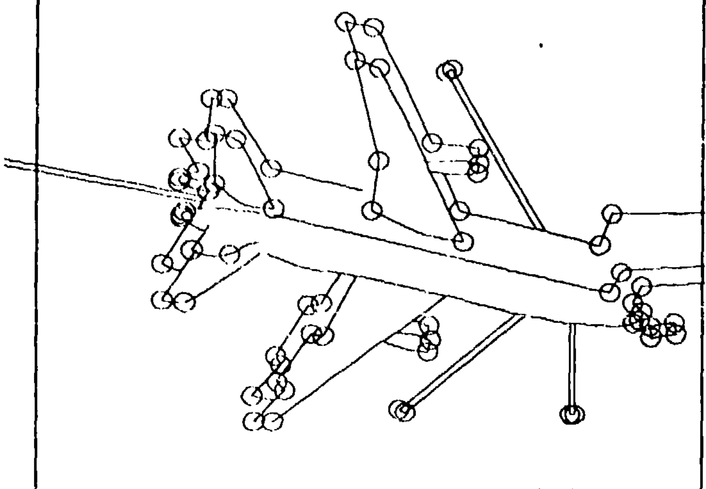


Fig 4 Hypotheses for boundary continuity.

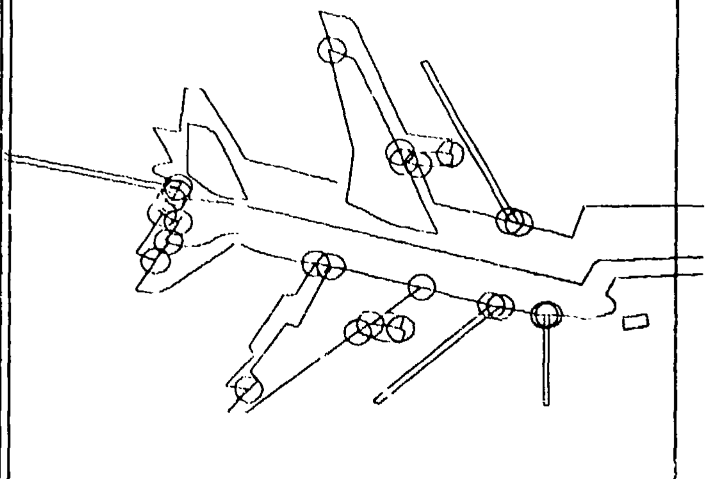


Fig 5 Terminations at continuous curves.

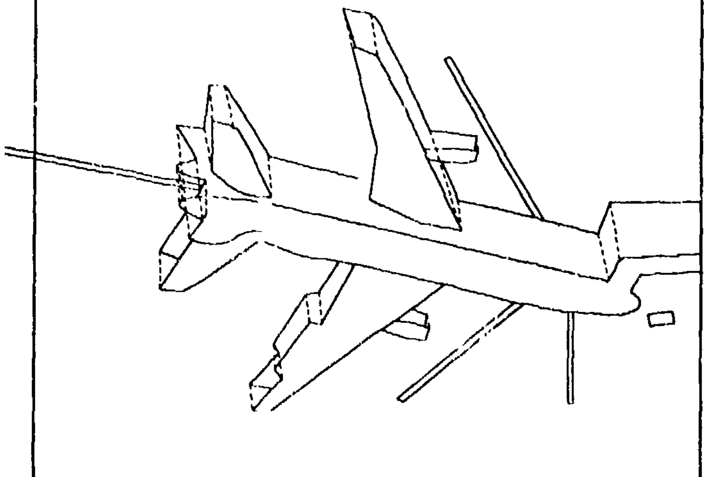


Fig 6 Hypothesised shadow breaks cast by edge breaks.

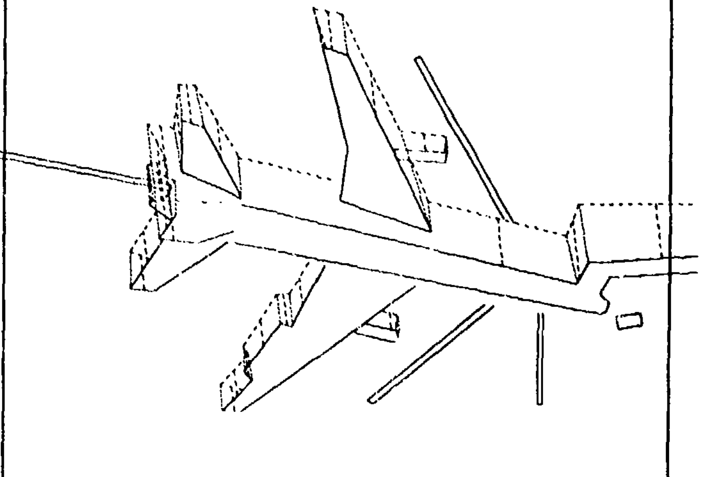


Fig 7 Match of shadows to geometric edges.

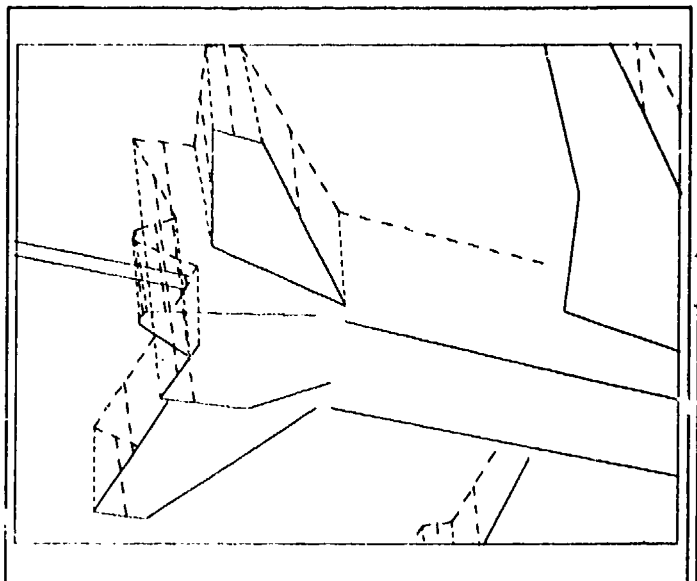


Fig 8 Closeup view of tail section shadow hypotheses.

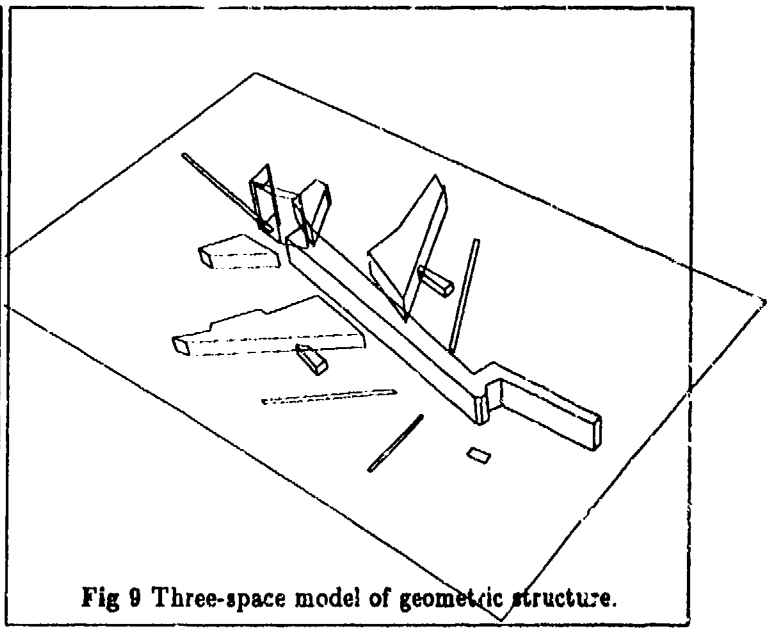


Fig 9 Three-space model of geometric structure.

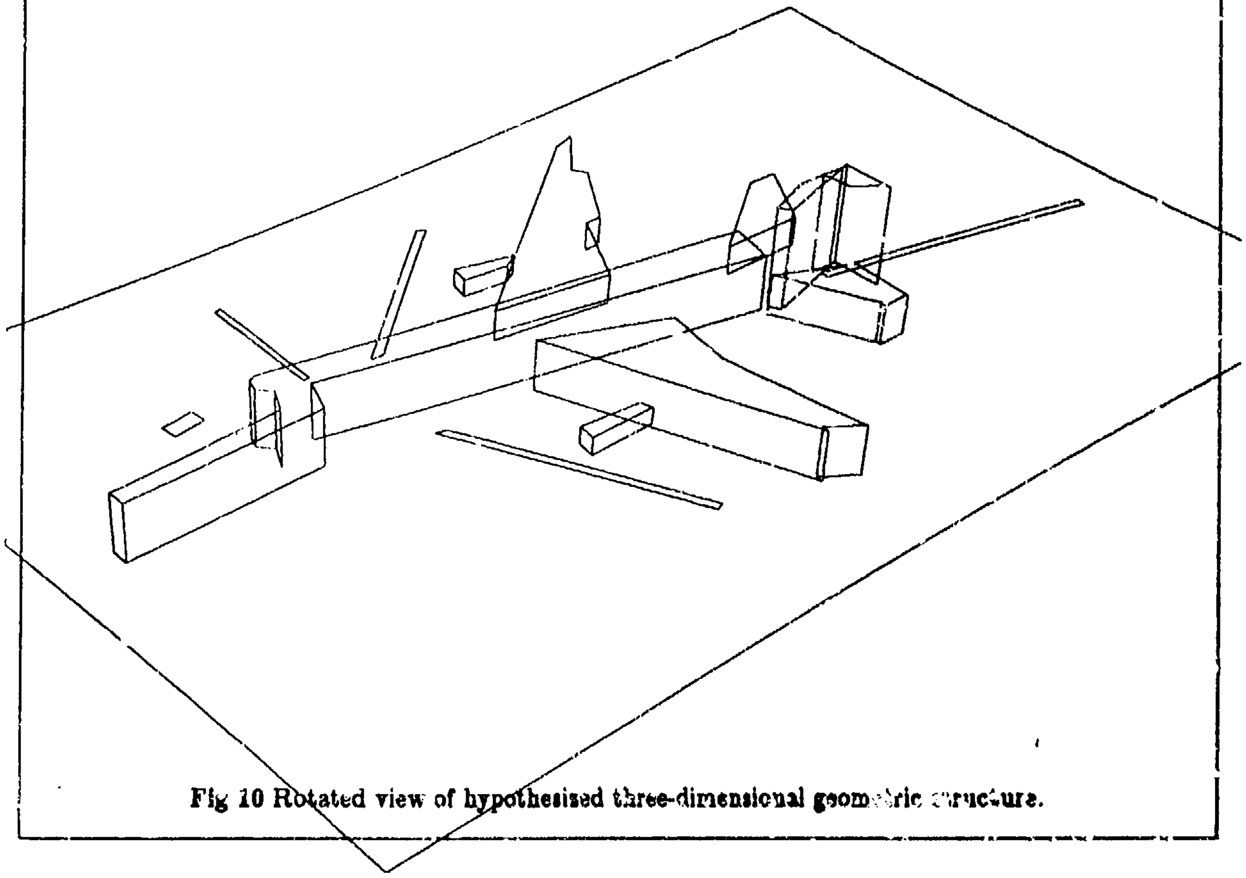


Fig 10 Rotated view of hypothesised three-dimensional geometric structure.

measurements are correct relative to some constant factor and can be made absolute by knowing the correct measurement for any one part of the image. Figures 9 and 10 show different views of such a model generated automatically from the hypotheses described above. Some of the minor errors in this model are actually artifacts of the display system rather than being present in the actual hypotheses. The position of the camera with respect to the ground was given to the modeling system, but this could be deduced from the image if the orientation of any image object with respect to the ground was known. With our digitized data it was impossible to see into shadows on the ground because of digitizing limitations, even though this information was available in the original negative. Interpretation would have been easier with more complete data

There is much more information available from the curve data than we have used in this example, and we hope to have better results soon. It is fairly easy to deduce the depth of objects suspended above the ground from the position of shadows which are cast underneath them, and we expect to be able to infer the cross-sections of objects in many cases from other clues. The interaction of these interpretations with knowledge of the specific objects in the scene is an important area for exploration.

Conclusions

The reasoning system used for this interpretation task can be summarized as follows: *we reason forward on the assumption that no coincidences have occurred, but always maintain enough information so that hypotheses can be reevaluated and reversed in the face of new contextual evidence.* This gives us the ability to use constraints which will sometimes be false, and therefore allows us to use many sources of information which would be unacceptable in a strictly deductive inference or constraint system. This method relies on the large amount of redundant information usually available for vision tasks. The further development of tools for reasoning with suggestive evidence is an important goal. Current programming techniques lack the flexibility needed to apply constraints of this sort in an optimal manner for different classes of input data.

One objective of this research has been to show that image intensity discontinuities carry far more information than has often been recognized in the past. The ability to precisely localize features is of great importance for providing strong constraints on interpretation. Not only is it valuable to localize intensity discontinuities as accurately as possible, but it is important to localize curve terminations, intersections, and tangent or curvature discontinuities accurately. The constraints on curve interpretations do not need to rely on restrictive assumptions about the class of objects in the image

More weakly localizable information, such as shading, intensity, and color, while being very important, is usually dependent on other assumptions regarding surface continuity. We believe that it is valuable to first examine the intensity discontinuities and carry out interpretation similar to that described here as a step towards using these other sources of information.

Shadows are not troublesome features to be removed from an image, but are in fact one of the most reliable sources of low-level information. While some of our constraints on shadows depend on properties of the sun and assume a known location of the light source, we believe that human perception in general makes use of the more qualitative aspects of our constraints on the interpretation of illumination. In particular, the behavior of illumination boundaries as they cross surface markings and geometric boundaries and terminate at geometric vertices provides a great deal of information even for unknown conditions of illumination. The identification of illumination boundaries can be carried out at a low level by noting the width of boundary transitions, constraints on contrast changes along a shadow edge, and the crossing of continuous curves.

Acknowledgements

Our task would have been much more difficult without the extensive systems and graphics features created by Rod Brooks during his work on ACRONYM. Sid Liebes has helped us work out the geometry of perspective imagery and shadow formation.

References

[1] Binford, Thomas O. "Inferring Surfaces from Images" *To be published in AI Journal*, 1981.

[2] Brooks, Rodney A., Russell Greiner and Thomas O. Binford. "The ACRONYM Model-Based Vision System," *Proc. of IJCAI-79*, Tokyo, Aug. 1979, 105-113.

[3] Clowes, M.B. "On seeing things," *Artificial Intelligence*, 2 (1971), 79-116.

[4] Erman, L.D. and V.R. Lesser "A Multi-level Organization for Problem Solving Using Many Diverse Cooperating Sources of Knowledge," *Proc. IJCAI-4*, (1975), 483-490.

[5] Huffman, D.A. "Impossible Objects as Nonsense Sentences," *Machine Intelligence*, 6 (1971), 295-323.

[6] Kanade, Takeo "A Theory of Origami World," *Artificial Intelligence*, 13 (1980), 279-311.

[7] Waltz, D. "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, P.H. Winston, (ed); McGraw-Hill, 1975.

RELATIVE DEPTH AND LOCAL SURFACE ORIENTATION FROM IMAGE MOTIONS

K. Prazdny

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

A simple mathematical formalism is presented suggesting a mechanism for computing relative depth of any two texture elements characterized by the same relative motion parameters. The method is based on a ratio of a function of the angular velocities of the projecting rays corresponding to the two texture elements. The angular velocity of a ray cannot, however, be computed directly from the instantaneous characterization of motion of a "retinal" point. It is shown how it can be obtained from the (linear) velocity of the image element on the projection surface and the first time derivative of its direction vector. A similar analysis produces a set of equations which directly yield local surface orientation relative to a given visual direction. The variables involved are scalar quantities directly measurable on the projection surface but, unlike the case of relative depth, the direction of (instantaneous) motion has to be computed by different means before the method can be applied. The relative merits of the two formalisms are briefly discussed.

The support of the Defense Advanced Research Projects Agency and the U S. Army Night Vision Laboratory under Contract DAAG-53-76-C-0138 (DARPA order 3206) is gratefully acknowledged, as is the help of Janet Salzman in preparing this paper.

1. INTRODUCTION

Optical flow is the distribution of angular velocities of projecting rays due to relative motion of objects with respect to the observer. Conceptually, optical flows undoubtedly carry a wealth of information about the spatial arrangement of the viewed scene, and prominent psychologists such as Gibson (1950, 1979) have argued forcefully for the predominant role of this information in human vision. Discontinuities in the distribution of angular velocities have been shown to directly correspond to occluding (or self-occluding) edges (Nakayama and Loomis, 1974). Corresponding discontinuities in "retinal" motions thus offer powerful information for segmentation purposes. Some recent work has illuminated some of the relationships between variables directly involved in the formation of optical flows (Koenderink and van Doorn, 1975, 1976, 1977; Longuet-Higgins and Prazdny, 1980; Prazdny, 1980).

The purpose of this paper is to present a mathematical analysis of some relations containing information about spatial dispositions of a set of texture elements. Using the concept of polar projection as the model for the physical image-forming process we show that the relative depth of two texture elements can be computed as a simple ratio. The entities involved are the angular velocities of the rays through the texture elements and the center of projection, and the visual directions of the rays, which are unit vectors specifying the directions of the rays in some egocentric reference frame centered at the center of projection. We show that the angular velocity at an image location can be obtained from the image velocity vector and its first time derivative at that locus.

A slightly different analysis requiring an a priori knowledge of the direction vector of the translatory component of the relative motion leads to an interesting characterization of local surface orientation.

Underlying our research in the interpretation of image motions is the assumption that (an approximation to) the velocity vectors associated with individual image elements can be obtained with reasonable accuracy. Some recent research regarding the computation of "retinal" velocities directly from the image brightness values (Hadani et al., 1980; Horn and Schunck, 1980) supports this assumption. Other promising support comes from the research on discrete solutions to the correspondence problem (Ullman, 1979; Barnard and Thompson, 1980) which relies on matching various higher-level image "token" structures.

2. THE BASIC IMAGE FORMING GEOMETRY

To begin, let us first consider motions of the projecting rays independently of any particular projection surface. Refer to Figure 1. A texture element P projects, along the ray OP, into the image element Q on the unit sphere centered at the center of projection, O. As the object moves relative to the observer, the ray OP (instantaneously) rotates about an axis through O causing the image element Q to trace a path (T3) on the unit sphere.

Consider a unit vector ($\bar{Q}$) determining the direction of the ray OP at a given instant <Note 1>. Its velocity is $d/dt(\bar{Q})=\dot{\bar{Q}}$. $\bar{Q}$ is perpendicular to $\dot{Q}$. As the ray moves with angular velocity $\underline{A}$, Q moves along T3 with (linear) velocity $\bar{Q}(=\underline{v}')$. The two velocities are related by $\dot{Q}=\underline{A}\times\bar{Q}$. Later we will show how the velocity $\underline{v}''$ of an image element on the projection plane relates to the velocity $\dot{\bar{Q}}$ of the unit vector $\bar{Q}$.

For the moment, it suffices to note that the equation $\dot{Q}=\underline{A}\times Q$ does not determine $\underline{A}$ uniquely; it only constrains $\underline{A}$ to lie in the plane a normal of which is $\dot{\bar{Q}}$ (see Figure 8 for further explanation).

The motions of the object and/or the observer are relative, i.e., we can only resolve the motion of the object relative to the observer (or vice versa). The (instantaneous) motion of an object with respect to the observer (in the reference frame in which the observer is stationary) can always be described as a rotation (with some angular velocity $\underline{A}_R$) superimposed on a translation specified by a vector $\underline{v}$. The axis of rotation can be chosen, without loss of generality, to pass through the center of projection (Chasles' theorem) to remove the ambiguity (Whittaker, 1944). The total (linear) velocity of an environmental point p (with position vector $\underline{P}$) is then $\underline{V}=\underline{v}+\underline{A}\times\underline{P}$. An equivalent expression in terms of the angular velocity of the projecting ray OP is

(1) $\underline{A}=\underline{A}_T+\underline{A}_R$

where $\underline{A}_T$ is the angular velocity due to the translation alone, and $\underline{A}_R$ is the rotational component (note that the values of $\underline{A}_R$ and $\underline{v}$ specify relative motion, and not, in any sense, the actual 3D motion of the object). The simplicity of equation (1) results from the fact that angular velocities about a common point add vectorially (Weatherburn, 1965). Observe that $\underline{A}_R$ does not vary from point to point on a rigid body; it is a property of the body as a whole and independent of distance or visual direction. $\underline{A}_T$, on the other hand, is a function of visual direction and the distance of the texture element to the center of projection (see below). Of these two component fields, only $\underline{A}_T$ carries information about relative distance. We will not derive an expression for $\underline{A}_T$ here. The reader is referred, for example, to Nakayama and Loomis (1974) or Prazdny (1980) for a detailed discussion. Briefly, when the object translates relative to the observer, individual rays of projection all move in the same plane (different for different rays). The direction of $\underline{A}_T$ is normal to this plane spanned by the vectors $\bar{v}$ and

$\overline{Q}$. The magnitude of $\underline{A}_T$ is equal to $d\beta/dt$ where β is the angle between the direction of translation and the given ray (see Figure 2). $\underline{A}_T$ is then given by

$$(2) \quad \underline{A}_T = d\beta/dt\ (\overline{v}\times\overline{Q}/\sin(\beta)) = v/S\ (\overline{v}\times\overline{Q})$$

where $S=|OP|$ is the distance of a given texture element to the center of projection.

The observation that enables us to derive an expression for the relative depth of any two texture points moving in the same way relative to the observer concerns equation (1). Consider any two points P_i, P_j on the same object <Note 2>. From equation (1) it follows that

$$(3) \quad \underline{A}_i = \underline{A}_{Ti} + \underline{A}_R \quad \text{and} \quad \underline{A}_j = \underline{A}_{Tj} + \underline{A}_R$$

We see that because $\underline{A}_R$ is the same for the two points it cancels out when the angular velocities are subtracted:

$$(4) \quad \underline{A}_{ij} = \underline{A}_i - \underline{A}_j = \underline{A}_{Ti} - \underline{A}_{Tj}$$

Using (2) and substituting we obtain

$$(5) \quad \underline{A}_{ij} = \overline{v}\times(k_i\overline{Q}_i - k_j\overline{Q}_j)$$

where $k_i = v/S_i$. We form the scalar product of both sides of (4) with $(k_i\overline{Q}_i - k_j\overline{Q}_j)$ to obtain

$$(6) \quad k_i(\underline{A}_{ij}\cdot\overline{Q}_i) - k_j(\underline{A}_{ij}\cdot\overline{Q}_j) = 0$$

This is because the scalar triple product involving one vector twice is always zero. We set $a_i = \underline{A}_{ij}\cdot\overline{Q}_i$ and substitute back into (5):

$$(7) \quad a_i/a_j = S_i/S_j$$

In other words, the relative depth of any two points having the same relative motion parameters ($\underline{v}$ and $\underline{A}_R$) is computable as a simple ratio. Observe that (6) does not involve $\underline{A}_R$ or $\underline{v}$; relative depth can be computed independently of relative motion. This is a simple but important finding. In general, all mathematical formalisms for computing surface orientation or 3D structure from motions on the projection surface depend (implicitly or explicitly) on computing the rotational component of the relative motion first (e.g., Ullman, 1979; Longuet-Higgins and Prazdny, 1980; Prazdny, 1980). To obtain $\underline{A}_R$, one has to solve a set of non-linear equations whose coefficients are the velocity vector components of at least five neighboring image elements (Prazdny, 1980), or use the first and second spatial derivatives of the image velocity field to obtain enough information to solve for $\underline{A}_R$ and $\underline{v}$ directly as an integral step in computing the local surface orientation (Longuet-Higgins and Prazdny, 1980). Ullman's scheme (Ullman, 1979), which uses orthogonal projection and relies on a theorem from affine geometry (the structure-from-motion theorem <Note 3>), uses not only spatial information (mutual position of a set of image elements) but also temporal information (the relative position of a given image element in successive snapshots) to recover the rotation of the configuration prior to the computation of relative depth [see Meiri (1980) for some comments on the number of image points and snapshots necessary to solve for the relative motion parameters]. Of course, the assumption of orthogonal projection works only for some situations.

In contrast, equation (6) above only requires that the angular velocities ($\underline{A}_i$, $\underline{A}_j$) at two visual directions ($\overline{Q}_i$, $\overline{Q}_j$) be known. Unfortunately, the angular velocity at a "retinal" locus cannot be computed from the information available at that locus at an instant. The equation $\underline{v}' = \underline{A}\times\overline{Q}$ does not specify $\underline{A}$ uniquely (see also Figure 8). The angular velocity of a ray through an image point can be obtained only when some additional information is available. We show that it can be computed when the vector specifying the time rate of change in the direction of the "retinal" velocity is available.

To see this consider Figure 3. Due to (relative) motion of a texture element, the ray specified by the direction vector $\overline{Q}$ rotates about O so that Q moves on the surface of the unit sphere with some velocity $\underline{v}' = \dot{\overline{Q}}$. To find the instantaneous plane of rotation of $\overline{Q}$ it is sufficient to apply a few concepts from elementary differential geometry. Observe that $\overline{v}'$, the unit vector in the direction of $\dot{\overline{Q}}$, is the unit tangent to the path at Q. This means that $\dot{\overline{v}}'$ is in the direction of the principal normal of Q. Together, $\bar{v}'$ and $\dot{\bar{v}}'$ span the plane on which lies the circle of curvature at Q. In other words, the plane a normal of which is $\bar{v}' \times \dot{\bar{v}}'$ (this vector lies in the direction of the binormal vector at Q) is the plane of instantaneous rotation of $\bar{Q}$, and $\underline{A}$, the angular velocity vector of Q, must lie in the direction of this vector. Observe that here we bring in temporal information to obtain the angular velocity vector. Other kinds of additional information are possible. In the next section we show, for completeness, how $\bar{v}'$ and $\dot{\bar{v}}'$ relate to "retinal" variables when the projection surface is a plane. Then we analyze a method for computing local surface orientation directly, without computing the relative depth map first.

3. COMPUTING THE ANGULAR VELOCITY OF A PROJECTION RAY FROM "RETINAL" VARIABLES

In this section we assume that the projection surface is a plane at unit distance from O. As the projecting ray rotates about O with some angular velocity $\underline{A}$, Q moves with velocity $\underline{v}'$ along T3, and $\underline{Q} = Q\bar{Q}$, the point at which the ray pierces the projection plane, moves with velocity $\underline{v}''$ along T2 (see Figure 4). Observe also that T3 is a perspective transformation of T2 and vice versa. For example, if T3 is a circle, T2 could be any conic section. The exact type of the curve will depend on the mutual orientation of the plane containing T3 (determined by the direction of the angular velocity vector $\underline{A}$) and PP. As mentioned above, the angular velocity vector $\underline{A}$ is normal to the instantaneous plane of rotation of the ray OP, i.e., parallel to the binormal vector. Our first task is thus to determine the direction of the binormal vector associated with the motion of $\bar{Q}$. First, we express $\dot{\bar{Q}}=\underline{v}'$ in terms of the velocity of the image element at a given point, and then we will find the direction of $\underline{A}$ as the vector product of $\bar{v}'$ and $\dot{\bar{v}}'$. We will establish a few interesting auxiliary relations before proceeding further.

Consider Figure 5. The figure illustrates the fact that the projection of a segment with magnitude v' along a perpendicular to a given line ℓ_2 which makes an angle λ with line ℓ is v" = v'Q/(sin(λ)-v'cos(λ)). We will refer to this equation as the "radial projection equation." Next we will establish a rather surprising fact about the relation between the velocity of a point and the velocity of its projection. We will first consider, for simplicity, only planar motions, but the relation holds for space motions too (see below). Consider Figure 6. Suppose that the point 'a' moves along a circular trajectory C so that its distance to O remains unchanged. The (infinitesimal) displacement is $d\varphi$. The displacement of the projection of 'a' on ℓ_1 is $\tan(d\varphi)$. The displacement of the projection of 'a' on ℓ_2 is $Q\tan(d\varphi)/[\sin(\lambda)-\cos(\lambda)\tan(d\varphi)]$ (using the radial projection equation). To compute the relation between the velocities on C and ℓ_2, we divide by dt and take the limit as $dt \to 0$:

$$\lim_{dt\to 0}\left[\frac{Q\tan(d\varphi)}{\sin(\lambda)-\cos(\lambda)\tan(d\varphi)}\ \frac{1}{dt}\right]$$

$$= \dot{\varphi}\ \frac{Q}{\sin(\lambda)}$$

This is because $\varphi(t)$ is a continuous function of t, and $\lim[\tan(x)/x]=1$ as $x\to 0$. This means that when a point 'a' moves along a path with speed v', its projection on the line ℓ_2 moves with speed

(8) $v''=Qv'/\sin(\lambda)$

In other words, the velocity of the projection of a point moving along a curve is <u>not</u> the projection of the velocity with which the point moves along that curve <Note 10>. This relation holds also in 3D space <Note 4>. Next we will derive the equation which will enable us to express the angular velocity in terms of the image velocities.

Refer to Figure 4. As $\bar{Q}$ moves along T3, its projection $Q\bar{Q}$ on PP moves along T2 with velocity

(9) $v''=d/dt(Q\bar{Q})=\dot{Q}\bar{Q}+Q\dot{\bar{Q}}=\dot{Q}\bar{Q}+Q\underline{v}'$

along T2. Observe that $\underline{v}''$, $\underline{v}'$, and $\bar{Q}$ all lie in the same plane <Note 4>. This means, however, that the following two vector equations hold simultaneously:

(10) $\bar{v}' \times \bar{Q}=(\bar{v}'' \times \bar{Q})/\sin(\lambda)$ and $\bar{v}'.\bar{Q}=0$

We can now solve for v' from these two simultaneous vector equations <Note 6> to obtain

(11) $\bar{v}'=\bar{Q}\times(\bar{v}''\times\bar{Q})/\sin(\lambda)=\csc(\lambda)\bar{v}''-\cot(\lambda)\bar{Q}$

This is the main equation. Note that $\bar{v}'$ is the unit tangent to T2 at $\bar{Q}$. Its first time derivative, $\dot{\bar{v}}'$, thus lies along the principal normal to T3 at $\bar{Q}$.

Their vector product in turn specifies the direction of the sought angular velrcity vector $\underline{A}$. Differentiating (11) we obtain

(12) $\dot{\bar{v}}'=\csc(\lambda)\dot{\bar{v}}''=\dot{\lambda}\csc(\lambda)\{\cos(\lambda)\bar{v}''-\bar{Q}\}-\cot(\lambda)v'$

Taking the vector product of (12) with $\underline{v}'$ and using the relation $\bar{v}\times\bar{v}=\cos(\lambda)(\bar{v}'\times\bar{Q})$ leads to

(13) $\bar{v}\times\dot{\bar{v}}'=\csc(\lambda)(\bar{v}'\times\dot{\bar{v}}'')+\dot{\lambda}(\bar{v}'\times\bar{Q})$

Substituting for $\bar{v}'$ from (11), for v' from (8), and simplifying, we finally obtain

(14) $\bar{v}'\times\dot{\bar{v}}'=Q\csc(\lambda)^2(\bar{v}''-\cos(\lambda)Q)\times\dot{\bar{v}}''+Q\dot{\lambda}\csc(\lambda)(\bar{v}''\times\bar{Q})$

Now $\dot{\lambda}=d/dt(\lambda)$ can be expressed in terms of $\dot{\bar{v}}''$ and the relation between the visual direction $\bar{Q}$ and the projection plane PP (its unit normal). Using $\bar{v}'.\bar{Q}=\cos(\lambda)$ and differentiating we obtain

(15) $\dot{\lambda}=-\csc(\lambda)(\dot{\bar{v}}''.\bar{Q}+\bar{v}''.\dot{\bar{Q}})=-\csc(\lambda)(\dot{\bar{v}}''.\bar{Q}+\bar{v}''.\dot{\bar{v}}')$

However, $\bar{v}''.\underline{v}'=(\bar{v}'\sin(\lambda)+Q\cos(\lambda)).v'=v'\sin(\lambda)$, because $\bar{v}'.Q=0$ by definition. Substituting for v' from (8) yields

(16) $\dot{\lambda}=-\csc(\lambda)(\dot{\bar{v}}''.Q)=v''\sin(\lambda)/Q$

Equations (14) and (16) indicate that the direction of the angular velocity vector at a visual direction $\bar{Q}$ can be determined completely once the image velocity vector $\underline{v}''$ at the locus on PP corresponding to the visual direction $\bar{Q}$, and the first time derivative of the direction vector of $\bar{v}''$, are available. The only other quantities entering the equation are Q and λ, expressing the metrics of the projective system <Note 6>.

Observe that the direction of the vector $\dot{\bar{v}}''$ is known as soon as $\underline{v}''$ is known. Because T2 is a planar motion, $\dot{\bar{v}}''$ lies in PP, and is perpendicular to $\underline{v}''$. Referring to Figure 7, we see that the direction of $\underline{v}''$, $\bar{v}''$, is specified by $\bar{v}''=\cos(\eta)\bar{x}+\sin(\eta)\bar{y}$ where $\bar{x}$ and $\bar{y}$ are a set of mutually perpendicular unit vectors on PP. $\dot{\bar{v}}''$ is thus given by

(17) $\dot{\bar{v}}''=\dot{\eta}[-\sin(\eta)\bar{x}+\cos(\eta)\bar{y}]$

and the magnitude of $\dot{\bar{v}}''$ is $\dot{\eta}$.

Finally, it remains to specify the magnitude of $\underline{A}$. Observe that A, the magnitude of $\underline{A}$, is a function of the mutual orientation of $\underline{A}$ and the direction vector $\bar{Q}$. This is related to the already mentioned fact that $\underline{v}'$ and $\bar{Q}$ do not specify $\underline{A}$ uniquely (see Figure 8). Using $\underline{v}'=\underline{A}\times\bar{Q}$ we see that $\underline{v}'=v'\bar{v}'=A(\bar{A}\times\bar{Q})$ $=A\sin(\omega)v'$, and from this it follows directly that

(18) $A=v'/\sin(\omega)=v''\sin(\lambda)/[Q\sin(\omega)]$

Here, ω is the angle between $\underline{A}$ and $\bar{Q}$, and $\cos(\omega)=\bar{A}.\bar{Q}$.

4. COMPUTING LOCAL SURFACE ORIENTATION

In this section, we analyze a method of computing local surface orientation relative to a given visual direction. An interesting result will be that the directions of angular velocity vectors are not required explicitly. However, we cannot obtain something for nothing; the analysis requires an a priori knowledge of the (instantaneous) direction of motion (the direction of the translatory component of the relative motion).

First, let us express vectors in a Cartesian (rectangular) coordinate frame as a function of two angles α and β. Then, for a given visual direction $Q(\alpha,\beta)$, we can compute $\partial\underline{A}/\partial\alpha$ and $\partial\underline{A}/\partial\beta$. Using equation (3), it is easy to see that

(19) $\partial\underline{A}/\partial\alpha=\partial\underline{A}_T/\partial\alpha$ and $\partial\underline{A}/\partial\beta=\partial\underline{A}_T/\partial\beta$.

In other words, the information contained in the gradient of the angular velocity field in the given visual direction is equivalent to the information contained in the gradient of its translatory component. This is not surprising, for as we saw above, only the translatory component carries information about depth relations between the 3D texture elements.

Let us find expressions for $\partial\underline{A}_T/\partial\alpha$ and $\partial\underline{A}_T/\partial\beta$ and analyze them to see how local surface orientation is specified in these expressions. If α and β are chosen such that the vector corresponding to $\alpha=0$ and $\beta=0$ specifies the direction of the translatory component $\underline{v}$, we see that $\bar{A}_T$ does not change as we move on the plane α=const. Using this fact and differentiating (2) with respect to β yields

(20) $\partial\underline{A}_T/\partial\beta=\partial A_T/\partial\beta\ \bar{A}_T$

Now, $\dot{\beta}=v\sin(\beta)/S$ (see Figure 2), and $\partial A_T/\partial\beta=\partial\dot{\beta}/\partial\beta$. Thus,

(21) $\partial\dot{\beta}/\partial\beta=v\cos(\beta)/S-\dot{\beta}S_\beta/S$ where $S_\beta=\frac{\partial S}{\partial\beta}$

Multiplying both sides by $\tan(\beta)$, and recalling that $\dot{\beta}=v\sin(\beta)/S$, we see that

$\tan(\beta)\partial\dot{\beta}/\partial\beta=\dot{\beta}-\dot{\beta}\tan(\beta)\ S_\beta/S$ so that

(22) $S_\beta/S=-1/\dot{\beta}[\partial\dot{\beta}/\partial\beta]+\cot(\beta)$

To derive a similar expression for S_α/S requires a little more ingenuity. Differentiating equation (2) with respect to α yields

(23) $\partial A_T/\partial_\alpha=-\sin(\beta)\ [v/S][S_\alpha/S]\overline{A}_T+[v/S](\overline{v}\times\overline{Q}_\alpha)$

But we have also

(24) $\partial \underline{A}_T/\partial\alpha=\partial(A_T\overline{A}_T)/\partial\alpha=\partial A_T/\partial\alpha\overline{A}_T+A_T\partial\overline{A}_T/\partial\alpha=\partial\dot{\beta}/\partial\alpha A_T+\underline{X}$

where $\underline{X}$ is some vector which does not have to be specified in detail (for our purposes). It can be seen immediately that

(25) $\partial\dot{\beta}/\partial\alpha=-[S_\alpha/S]v\sin(\beta)/S$

and from this S_α/S is specified as

(26) $S_\alpha/S=-$[illegible]$\partial\dot{\beta}/\partial\alpha$

The quantities S_α/S and S_β/S are depth invariant characterizations of (local) surface orientation relative to a particular visual direction $\overline{Q}(\alpha,\beta)$ <Note 8>. In fact, they are directly related to the gradient of the distance. Because of the dependence of this specification of local surface orientation on a particular visual direction (defining the surfaces of constant α and β), two different surface orientations cannot be directly compared. To do so, one could transform one characterization into another using a simple rotation matrix. It is important to realize that the expressions characterizing the (local) surface orientation in a pure translatory situation hold also in a general situation of a curvilinear motion. The only prerequisite is (as in the pure translation case) that the direction of the instantaneous motion (i.e., the direction of the translatory component of the curvilinear motion) is known. The knowledge of $\overline{v}$ allows us to define, for each "retinal" locus, a direction along which α=const. By projecting the "retinal" velocity vector into this direction we obtain $\dot{\beta}$, and by differentiating these "retinal" velocities along the directions α =const. and β=const. (see Figure 10) we obtain $\partial\dot{\beta}/\partial\alpha$ and $\partial\dot{\beta}/\partial\beta$.

If this process is to be carried out on the projection plane, the best thing to do is to locate the focus of expansion which then determines the lines α=constant as the lines joining the focus of expansion and the particular "retinal" locus. The localization of the focus of expansion (FOE) is a difficult task. One may try to decompose the image velocity field there into its constituents. The rotational component of the angular velocity vector at a given instant can be decomposed into two image velocity fields: one, a hyperbolic field (the component vectors being tangent to hyperbolas through given loci) due to rotation about an axis through the center of projection parallel to the projection plane, and the other, a circular field (the component vectors being tangent to circles with centers at the center of the image coordinate system) due to rotation about the normal to the projection plane. The remaining translational component would consist of vectors defining the focus of expansion (possibly at infinity) as the unique intersection of all straight lines defined by these vectors and the corresponding "retinal" loci. The whole process is essentially a constrained minimalization problem of a function of three variables: the magnitude of the circular component, the direction of the hyperbolic field (specified by a single angle), and the magnitude of the hyperbolic field (see Figure 11). The constraining condition is that the straight lines of the translational component meet at FOE. Note that such a process, if successful, would essentially recover the translational as well as the rotational component of the angular velocity vector $\underline{A}$ (see equation (1)). We are currently trying to solve this problem using a relaxation scheme. <Note 9>.

5. CONCLUSION

In this paper we have shown that the relative depth of any two texture elements moving in the same way relative to the observer can be computed in a simple way from the angular velocities of the corresponding rays of projection. We have illustrated how the required angular velocity at a point on the planar projection surface can easily be computed from the (linear) velocity of the image element at that locus, and the first time derivative of its direction vector. We have also shown that local surface orientation can be obtained rather straightforwardly once the direction of the translatory component of the relative motion is known. The recovery of this direction from the information contained in the distribution of the "retinal" velocities is a rather complicated task. It is hoped that it may be possible to decompose the instantaneous velocity field on the projection plane into its constituents using a relaxation process. Some work on this problem is currently in progress in our laboratory.

The applicability of the method will depend on the accuracy with which the image velocities can be obtained. It remains to be specified how these errors will propagate through the equations and affect the accuracy of the computed relative depth and local surface orientation.

The computation of relative depth and local surface orientation were presented as two distinct processes. This does not have to be so. Local surface orientation may be obtained from a relative depth map, for example, by simply fitting a plane to a set of relative depth values in a given (small) neighborhood. I believe that the relative depth map is practically a much more useful construct than local surface orientation. Because the available data are noisy, the computation of local surface orientation relying on quantities obtained in a small neighborhood of a "retinal" point is likely to be affected much more than the relative depth of two widely separated points, where the two angular velocities can be obtained much more precisely, e.g., by averaging over a small neighborhood.

REFERENCES

1. Clocksin, W. F., 1980, Perception of surface slant and edge labels from optical flow: a computational approach. Perception 9, 253-267.

2. Hadani, I., Ishai, G., Gur, M., 1980, Visual stability and space perception in monocular vision. J. Opt. Soc. Am. 70, 60-65.

3. Horn, B. K. P., Schunck, B. G., 1980, Determining optical flow. MIT AI Memo No. 572, Massachusetts Institute of Technology.

4. Gibson, J. J., 1950, The perception of the Visual World. Boston: Houghton-Mifflin.

5. Gibson, J. J., 1979, The Ecological Approach to Visual Perception. Boston: Houghton-Mifflin.

6. Koenderink, J. J., van Doorn, A. J., 1975, Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. Optica Acta 22, 772-791.

7. Koenderink, J. J., van Doorn, A. J., 1976, Local structure of movement parallax of the plane. J. Opt. Soc. Am. 66, 717-723.

8. Koenderink, J. J., van Doorn, A. J., 1977, How an ambulant observer can construct a model of the environment from the geometrical structure of the visual inflow. In Hauske and Butenandt (eds.), Kybernetik, pp. 224-247, Munich: Oldenburg.

9. Longuet-Higgins, C., Prazdny, K., 1980, The interpretation of a moving retinal image. Proc. R. Soc. London B-208, 385-397.

10. Meiri, A. Z., 1980, On monocular perception of 3-D moving objects. IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-2, 582-583.

11. Prazdny, K., 1980, Egomotion and relative depth map from optical flow. Biol. Cybernetics 36, 87-102.

12. Weatherburn, C. E., 1965, Elementary Vector Analysis. London: Bell & Sons.

13. Whittaker, E. T., 1944, A treatise on the Analytical Dynamics of Particles and Rigid Bodies. New York: Dover.

NOTES

<Note 1>

The following notation is used through the paper. If $\underline{v}$ is a vector then $\bar{v}$ is a unit vector in its direction, and v is the magnitude of $\underline{v}$. The scalar product is denoted by ".", and the vector product by "X". All velocities, position vectors, and the associated quantities are functions of time. This is assumed implicitly throughout the paper. $\dot{\underline{v}}$ and $\dot{v}$ denote the time derivatives, as usual. Angular velocities are vectors perpendicular to the plane of (instantaneous) rotation, with magnitude equal to angular speed (radians/sec).

<Note 2>

The texture elements can be on two different objects as long as the objects move in the same way relatively to the observer (i.e., have the same $\underline{v}$ and $\underline{A_R}$). Thus, for example, in the stationary world where the observer is the only moving agent, the relative depth of all texture elements can be recovered using the present method.

<Note 3>

The structure-from-motion theorem states that the relative depth of four non-coplanar points is recoverable from three non-degenerate orthographic projections. The mutual orientation of the projection planes has to be determined before the actual relative depth of the four points can be computed. The recovery of the mutual orientation of the projection planes is an integral part of the schema.

<Note 4>

To see this, note that $\underline{v}''=d/dt(Q\overline{Q})=\dot{Q}\overline{Q}+Q\dot{\overline{Q}}$

Now $\dot{\overline{Q}}=\underline{v}'$ and $\underline{v}'.\overline{Q}=0$. Thus $\underline{v}''\times\overline{Q}=Q(\underline{v}'\times\overline{Q})$, i.e., $\underline{v}'$, $\underline{v}''$, and $\overline{Q}$ all lie in the same plane. Setting $\overline{n}$ to be the unit normal of this plane, we have $\underline{v}'\times\overline{Q}=v'\overline{n}$; but also $\underline{v}''\times\overline{Q}=v''\sin(\lambda)\overline{n}$ (see Figure 4). Thus $\overline{n}=(\underline{v}'\times\overline{Q})/v'=(\underline{v}''\times\overline{Q})/v''\sin(\lambda)$. Substituting for $\underline{v}''\times\overline{Q}$ we obtain $(\underline{v}'\times\overline{Q})/v'=Q(\underline{v}'\times\overline{Q})/v''\sin(\lambda)$ and so $v'=v''\sin(\lambda)/Q$, as stated in (8).

<Note 5>

A set of vector equation of the form

$$\underline{X}\times\underline{A}=\underline{B} \text{ and } \underline{X}.\underline{C}=p$$

where $\underline{A}.\underline{C}\neq 0$, has a general solution $\underline{X}=(p\underline{A}+\underline{C}\times\underline{B})/(\underline{A}.\underline{C})$.

<Note 6>

Given the unit normal $\overline{m}$ defining the projection plane PP, these quantities are computable easily as

$$Q=1/(\overline{Q}.\overline{m}), \text{ and } \lambda \text{ is given by } \cos(\lambda)=(\overline{Q}.\overline{v}'').$$

<Note 7>

As can be seen in Figure 9, $\overline{A}_T$ is the same for all $\overline{Q}(\alpha,\beta)$ on the plane α=const.; it is the unit normal of this plane. It follows directly that $\frac{\partial\overline{A}_T}{\partial\beta}=0$.

<Note 8>

Expressions similar to (22) and (26) were also independently obtained by Clocksin (1980) using a different approach.

<Note 9>

While the problem is conceptually rather simple, there are some difficulties relating to the formulation of the criterion function to be minimized. The difficulty is related to the fact that the projection plane is an augmented Euclidean plane, in terms of projective geometry. For example, the translational vector components on the plane may all meet at an (ideal) point at infinity. It is rather difficult to incorporate this condition into a nicely behaving criterion function.

<Note 10>

This result is intuitively rather surprising. It follows directly, however, from the definition of the angular velocity (see Figure 2).

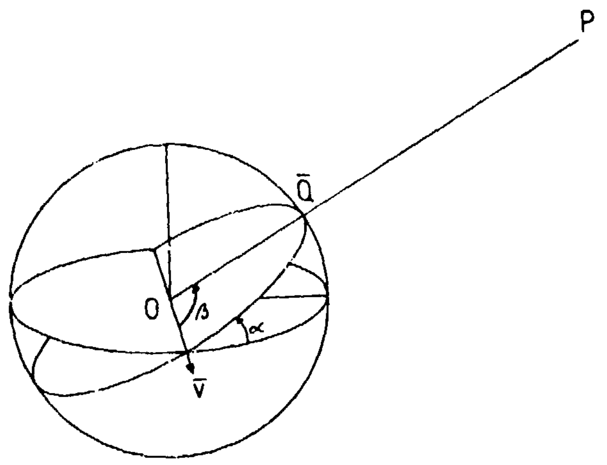


Figure 1. A texture point P projects into a point Q on the unit sphere. The direction vector of the projecting ray OP is determined by the two angles, α, the meridian, and β, the eccentricity; the vector $\bar{Q}$ is a function of α and β. The plane α=0 and the direction (α=0,β=0) are arbitrary, but it is advantageous for the future analysis to choose them so that the principal x-axis coincides with the direction vector of the translatory motion component.

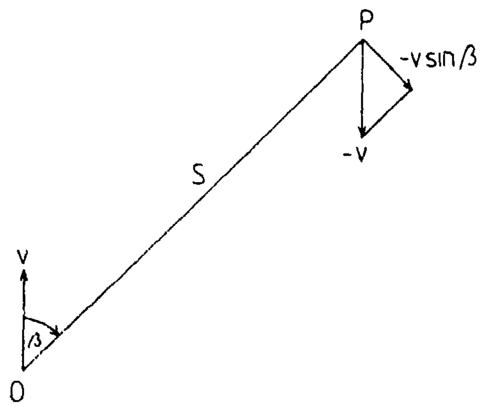


Figure 2. The angular velocity of a ray due to a pure translation. The angular speed is defined as $d\beta/dt$, i.e., as the projection of v on the perpendicular to the ray, divided by the distance S.

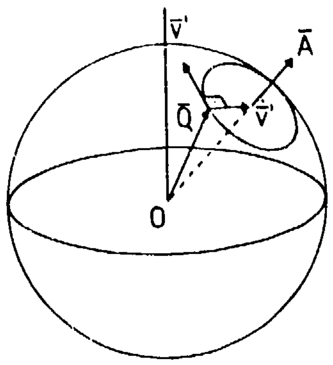


Figure 3. To compute the direction of the angular velocity vector of the ray specified by the visual direction $\bar{Q}$, observe that $\bar{Q}$ moves (on a 3D path on the surface of the unit sphere) with velocity $\underline{v}'=\dot{\bar{Q}}$. Because $\bar{v}'$ is the unit tangent to this path, $\bar{v}$ lies in the direction of the principal normal to the path at $\bar{Q}$. The unit binormal vector at $\bar{Q}$, which is perpendicular to the plane spanned by $\bar{v}'$ and $\dot{\bar{v}}'$, is parallel to the angular velocity vector $\underline{A}$ of $\bar{Q}$.

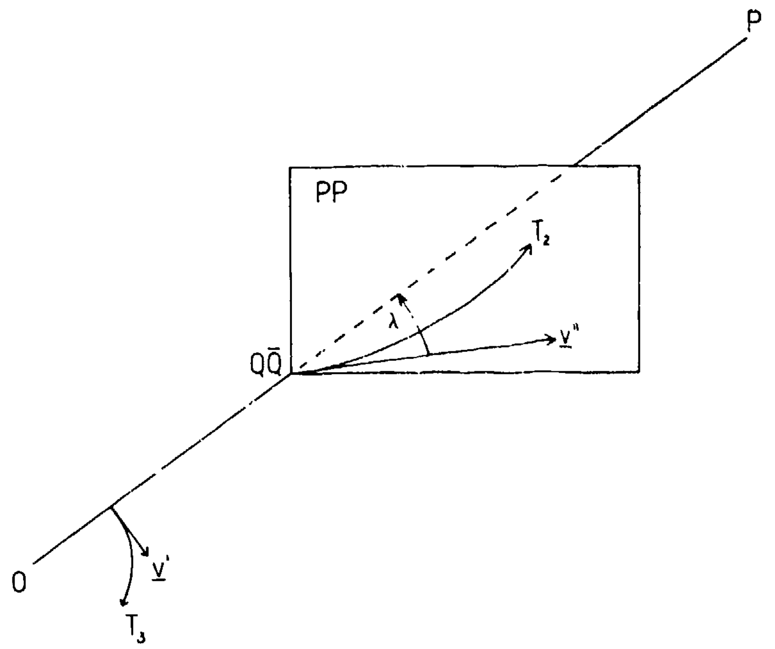


Figure 4. The basic projection geometry. The ray, determined by its direction vector $\bar{Q}$, moves due to the relative motion of the object with respect to the observer. The point $Q\bar{Q}=\underline{Q}$ at which it pierces the planar projection surface, PP, describes a planar trajectory T2. The unit vector $\bar{Q}$ describes a 3D trajectory T3. The angle λ is the angle between the image velocity $\underline{v}''$ of $\underline{Q}$, the projection of P onto PP, and the ray OP.

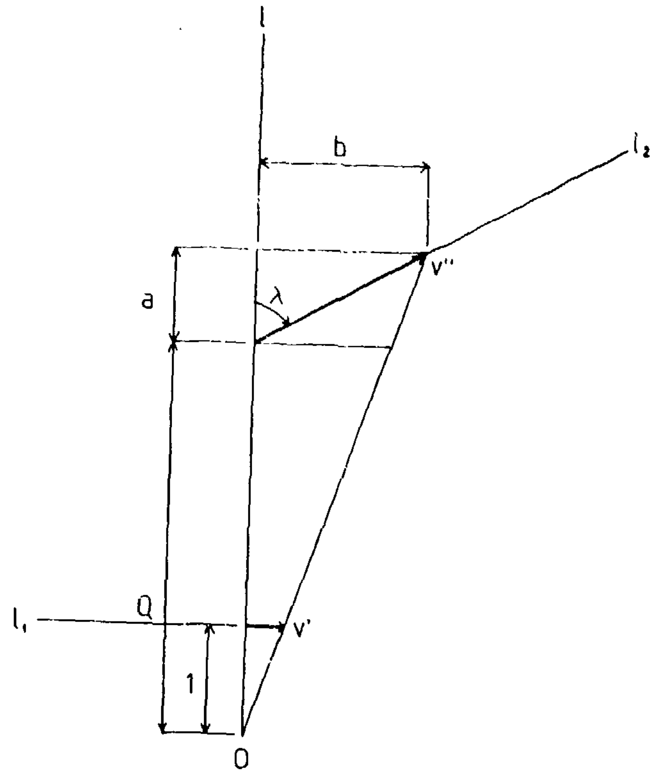
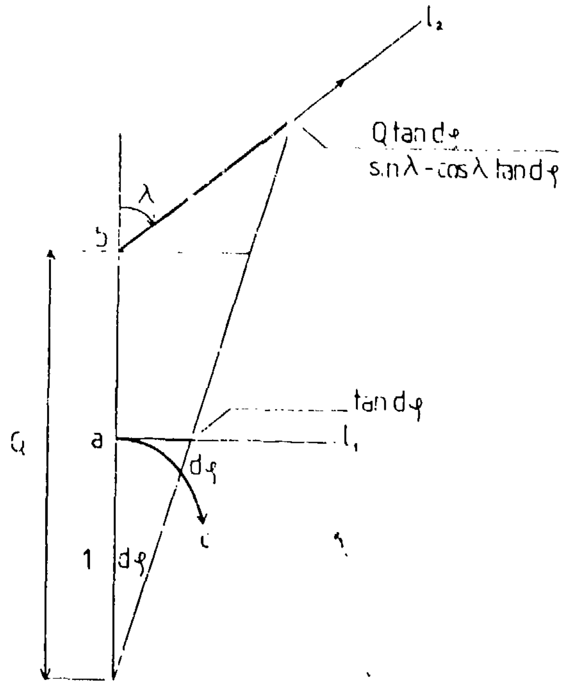


Figure 5. The radial projection equation. The displacement v' on ℓ_1 at unit distance from the center of projection is projected into the displacement v" on ℓ_2. To compute the relation between v' and v", we note that b/(a+Q)=v'; b/v"=sin(λ); and a/v"=cos(λ). Thus v"sin(λ)=v'v"cos(λ)+v'Q. Finally, v"=v'Q/(sin(λ)-v'cos(λ)). Observe that here, v' and v" are finite displacements and not velocity vectors!



l_2
$\frac{Q \tan d\varphi}{\sin\lambda - \cos\lambda \tan d\varphi}$
λ
b
$\tan d\varphi$
Q
a
l_1
$d\varphi$
C
1
$d\varphi$

Figure 6. The relation between an infinitesimal displacement $d\varphi$ along the circle C, and its projection on the line ℓ_2. The speed at which b, the projection of a on ℓ_2, moves along ℓ_2 is not the projection of the speed with which a moves along C. See text for further explanation.

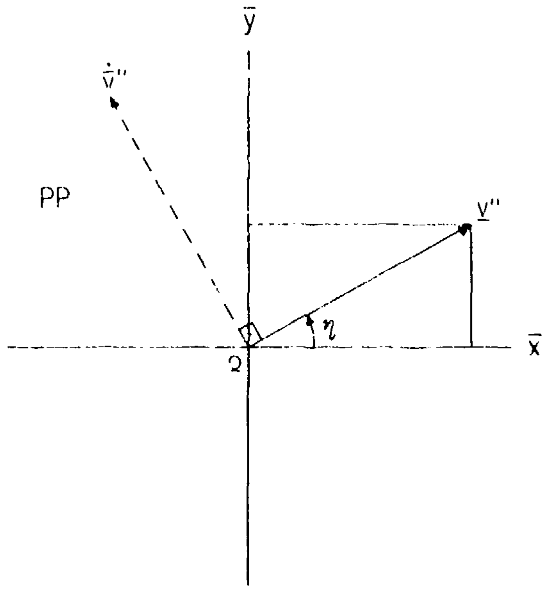


Figure 7. The direction of $\underline{v}''$ on PP is determined by an angle η. The first time derivative of $\bar{v}''$ has direction perpendicular to $\bar{v}''$, and magnitude $\dot{\eta}$. $\bar{x}$ and $\bar{y}$ are a set of mutually perpendicular unit vectors on the projection plane PP.

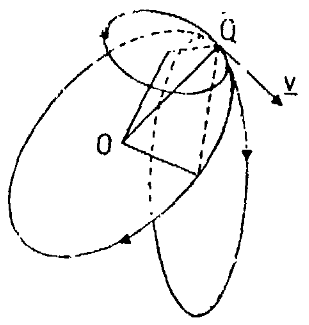


Figure 8. Knowing that the point Q (with position vector $\bar{Q}$) moves with some (linear) velocity $\underline{v}$ does not specify the angular velocity of the ray OQ. The equation $\underline{v}=\underline{A}\times\bar{Q}$ constrains $\underline{A}$ to lie in the plane of which v is a normal. Q could (instantaneously) move on an infinite number of possible circles of rotation, only three of them being shown. Observe that $\underline{A}$, the magnitude of $\underline{A}$, depends on the angle between $\bar{Q}$ and $\bar{A}$ (it determines the radius of the instantaneous circle of rotation).

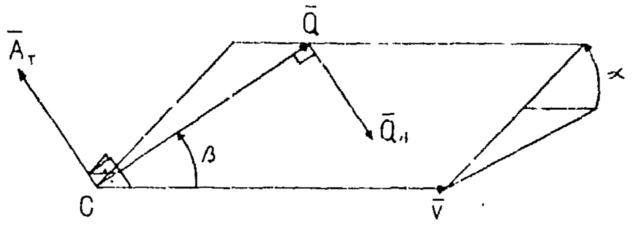


Figure 9. $\bar{A}_T \perp \bar{Q} \wedge \bar{A}_T \perp \bar{Q}_\beta$.

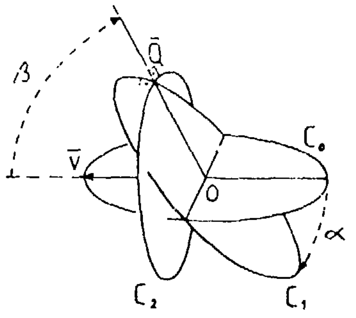


Figure 10. For each given visual direction $\bar{Q}(\alpha,\beta)$, the circles α=const., β=const., and $\bar{Q}$ itself define a rectangular coordinate system. Observe that while the condition α=const. defines a plane, β=const defines a cone with apex at O! The circles C_1 on the plane and C_2 on the cone are mutually perpendicular.

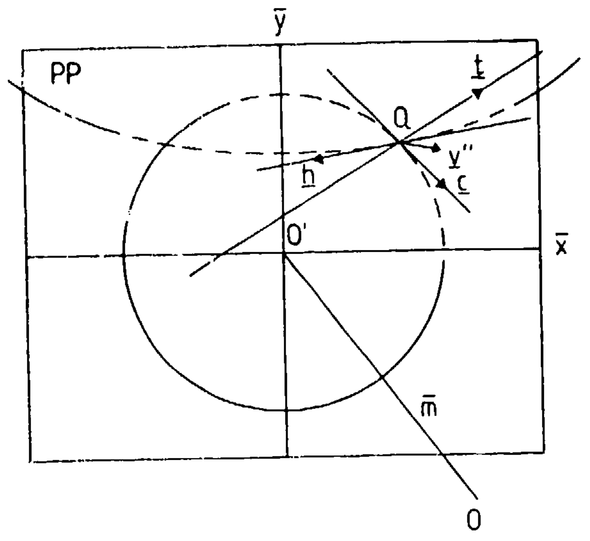


Figure 11. An image velocity $\underline{v}''$ (on the planar projection sur-
face PP) of a point Q can be resolved into three compon nts. The
hyperbolic component $\underline{h}$ is due to the rotation of the ray about an
axis (through O) in the projection plane (the angular velocity
is a linear combination of $\bar{x}$ and $\bar{y}$). The circular component $\underline{c}$
is due to the rotation of the ray about an axis (through O) paral-
lel to $\bar{m}$. The translational component $\underline{t}$ is the remaining vector
which constraints the decomposition of $\underline{v}''$; $\underline{t}$ is constrained to be
such that $\forall Q_i \in PP$: ($\underline{t}_i$ intersect in one common point). In the
illustration above, the direction angle of the hyperbolic field
is zero (measured anticlockwise from the x-axis).

STRUCTURAL ANALYSIS OF NATURAL TEXTURES*

Felicia Vilnrotter
Ramakant Nevatia
Keith E. Price

Image Processing Institute
University of Southern California
Los Angeles, California 90007

ABSTRACT

A technique to analyze patterns in terms of individual texture primitives and their spatial relationships is described. The technique is applied to natural textures. The descriptions consist of the primitive sizes and their repetition pattern if any. The derived descriptions can be used for recognition or reconstruction of the pattern.

INTRODUCTION

Areas of an image are better characterized by descriptions of their texture than by pure intensity information. Texture is most easily described as the pattern of the spatial arrangement of different intensities (or colors). The different textures in an image are usually very apparent to a human observer, but automatic description of these patterns has proved to be very complex. In this research, we are concerned with a description of the texture which corresponds, in some sense, to a description produced by a person looking at the image.

Many statistical textural measures have been proposed in the past [1-4]. Reference 1 gives a good review of various texture analysis methods. Among the statistical measures which have been discussed, and used, analysis of generalized gray-level co-occurrence matrices [2], analysis of edge directions with co-occurrence matrices [3], and analysis of the edges (or micro-edges) in a subwindow [4]. The statistical methods, by themselves, do not produce descriptions in the form which we desire. Some of the measures may indicate certain underlying structures in the pattern, but do not produce a general description. The Fourier transform has been used to determine some structural descriptions but was only partially successful for more complex patterns [5].

The work in what can be called structural texture description has been more limited [6-9]. Maleson [6] used simple regions as the basic elements and used relations between regions and shape properties of the region in his analysis. Rosenfeld [7] has proposed a texture analysis method also based on primitive regions extracted by a threshold operation. Tamura et al. [8] tried to develop a set of operators which would rate textures on several scales, comparable to their ratings by human subjects. The proposals of Marr [9] for texture analysis based on the primal sketch are similar to some of the analysis which we perform.

This paper builds on work reported earlier [10,11] and thus we will only outline the early processing steps. We will present more detail on determining spatial relations between primitives and generation of textures from the descriptions. Finally, we comment on the use of the descriptions for recognition and the computation of texture gradients.

SYMBOLIC DESCRIPTIONS

The goal of this work is a description of a texture pattern based on primitive elements and the spatial arrangement of those elements. For example, a checker board pattern might be described as approximately square elements, alternately light and dark in perpendicular directions. Absolute intensity (color) information is an unreliable feature for defining texture primitives. Instead, we have chosen to analyze edge images to initially determine the structure.

The edge repetition arrays (ERAs) are similar to gray level co-occurrence matrices, but differ in many respects. The important feature which we wish to extract is the repetitive nature of the texture patterns. This is apparent when a sequence of spacings (e.g. 2-32) is considered. Thus the values which we wish to compute are how edge elements repeat as a function of distance for a given angle.

We accumulate edge repetition information (for edges the same direction, and edges opposite directions) for the six edge directions (0^o, 30^o, 60^o, etc.) and for both colors ("light" and

*This research is supported by Defense Advanced Research Projects Agency and is monitored by the Wright-Patterson Air Force Base under contract No. F33615-80-C-1080, DARPA Order No. 3119.

"dark") and for spacings of 2 to 32 or more. The data for a particular edge direction are accumulated by scanning the image in a direction perpendicular to the edge direction. The sums are normalized to give a value which is the probability of a pair of edges of the proper direction occurring at the given angle and spacing when one edge occurs.

Well defined texture patterns produce obvious patterns in the ERA, and it is from these patterns in the ERA which we will derive the description of the texture pattern. For example, a regular pattern of dark and light elements will produce an edge image with regularly spaced edges. The ERA for opposite edge directions will have a peak value at the element size and peaks spaced by the width of the two elements. A pattern created by the random arrangement of similar elements will not have significant numbers of edge repetitions (for opposite edge directions) except for the distance corresponding to the element width - one relatively large peak at a small distance and lower values afterward with no other dominant distance value. In the data for the same edge direction the situation is reversed - few repetitions at small distances and relatively more at larger distances, but no repetitive structure in the data.

The symbolic description of the pattern is derived from the edge repetition arrays. The basic technique is summarized below:

1) Find, classify and describe the peaks (local maxima) in the ERA's,i.e., as strong or weak.

2) Examine the element spacing (size of pairs of elements) data to determine if there are any peaks which repeat - i.e., occur at multiples of the first element spacing value.

3) Examine the element size data to determine if there is support for a spacing value determined above - i.e., local maxima spaced at the given distance.

4) Apply a set of heuristics, expressed as production rules, to generate the final description.

The description of the texture of Fig. 1 is given in Fig. 2.

DESCRIPTION OF PRIMITIVES

The basic symbolic description outlines in Section II above is only a start. There remain many problems. There is no indication of the overall shape of the elements, there is no coordination of description from one direction with descriptions for other directions, and there is no way to immediately derive various properties of the elements. To determine these properties we must extract all the primitive texture elements from the image. Earlier methods [6,7] made an attempt to extract primitives as the first step. But, there are problems when elements are very small, the intensity levels vary across the image or there are several different types of elements (e.g., three with different intensities) which compose the texture pattern. Therefore, instead of generating texture elements as a first step, we look for texture elements with certain known properties, the properties given in the basic symbolic description.

There are three strong indications of element size provided in the texture description given in Fig. 2. Knowing the exact locations, within the original texture image, where the edge matches contributing to these strong peaks occur is useful. It is then possible to isolate the uniform intensity regions or textural elements being measured. Analysis of the set of textural elements or primitives for a particular predominant element size then provides the average intensity, are, shape, etc. for that primitive group.

In Fig. 2, we can see that all of the information for a texture is listed according to relative element intensity and scan direction. For example, elements of size 3 and spacing 8 occur in the vertical scan direction; and these elements are dark in relation to their vertical neighbors. Since the scan direction and relative element intensity of each predominate element size is known, the edge pairs exhibiting the properties can be located.

The points between these pairs of edges serve as the initial interior points of the texture elements. These primitive slices are expanded to form the mask for the particular primitive elements. The original image and the binary edge image are both used by the expansion procedure. The expansion is perpendicular to the scan direction, i.e., if the initial description is for the vertical then the expansion is in the horizontal direction.

The expanded primitives within the mask image can then be analyzed to determine various properties of the basic element such as an average primitive intensity, an average primitive area in pixels and the average primitive dimension in the direction perpendicular to the line of scan.

Results generated for the raffia sample of Fig. 1 are given in Fig. 3. The dark primitive found in the vertical scan direction corresponds to bock B in the abstract representation of the texture pattern given in Fig. 4. The light primitive found in the vertical scan direction corresponds to block A in Fig. 4 and the light primitive found by the horizontal scan corresponds to block C. The first primitive dimension given in each of the primitive descriptions is the dimension along the line of scan, (i.e., the direction in which it was described earlier in Fig. 2 while the second dimension is in a direction perpendicular to the scan line. Figure 5 gives the primitive mask images corresponding to the primitives of types.

SPATIAL RELATIONS OF TEXTURE ELEMENTS

Up to this point the only description information we have relating to the arrangement of the texture primitives is the element spacing information described in Section II. However, this information pertains to only a single scan direction. Therefore, we can tell only if a certain group of primitives in periodic in one direction and if so, what period is exhibited. When no element spacing can be found for any of the elements within a texture, the texture pattern is assumed to be random. In the event that we are considering a non-random texture pattern this spacing information is not sufficient to characterize the particular placement rules within the texture. There is other work in the area of describing textures by relationships between elements [12,13,14] but these others generally make more assumptions about the type of input textures.

Consider the 2 brick patterns in Fig. 6 (a and b). These patterns are similar in that each contains light and dark elements which are rectangular and are arranged in a regular pattern. The arrangement of the bricks within the patterns is different, but no evidence of this difference is given in their primitive descriptions (see Fig. 7a and b). Both sets of bricks were detected by scanning vertically, and both exhibit regular element spacing in this direction. However, a definitive description of element arrangement is lacking. This placement information is contained in the primitive masks and composite images (see Fig. 8), and is the object of analysis described here.

Determination of Relations

Individual primitive masks provide the locations of all the primitives of a particular type within an image. Determining the predominant placement rules exhibited by these elements can be divided into a number of independent subtasks:

1) Determine whether 2 sets of primitive masks represent the same textural elements by combining all pairs of masks to calculate overlap. This situation arises when a textural element is detected in more than one scan direction. This should give a reduced set of primitives.

2) Determine primitive location (centroid) and compute relations. Rather than computing relations by taking all pairs, only the relations to primitives (of any type) closest to a given primitive are considered. The closest ones are located by looking in 12 directions (30^{O} apart) from the given primitive. Totals are kept for angle, distance, and the type of pair.

3) Find the predominant relations: Normalized the values to adjust for numbers of the type of primitives. Replace all points above a threshold by the sum of their 3x3 neighborhood. Find the predominate local maxima. This gives a pair of primitives and an angle and distance separating them. Figure 9 gives the computed results for the two brick patterns. This can serve as a basis for descriptions or it can be used to reconstruct the texture.

Generation of Patterns

One indication of the completeness of a texture description is the effectiveness the description to regenerate the texture pattern. The descriptions given in Fig. 9 (and a description of each element) can be used to recreate the original texture patterns. Certain assumptions about the pattern are necessary: it is a single pattern (composed of all the derived element types) and the relations between elements of the same type is the same for all types of elements (i.e., if a grid is found for placing one element type, the same grid can be used for all others - with translation).

The generation process has several steps:

1) Ratings for relations which are the same, except for a 180^{O} difference in directions, are combined.

2) The highest rated relation for an element with itself is chosen (marked with "*" in Fig. 9). But itself, this gives only a one dimensional pattern, therefore the second best relation must be also used (marked with "**"). Collinear relations are ignored and the rating is dependent on the distance - relations to close objects are needed for generation of the pattern. Thus in Fig. 9b the 150^{O}-30^{O} pair is chosen rather than the 180^{O}, 0^{O} pair. These two placement rules define a basic grid for the pattern (i.e., place an element at each grid point).

3) Select the highest rated relation between an element already in the grid and one not in the grid (marked with "***" in Fig. 9). Put this element into the pattern in the given relation to all those already there.

4) Continue to 3 for all other elements (the examples here have only 2 elements, but 3 or more are possible).

Figure 10 shows the results for these 2 patterns. In the first pattern some of the mortar area was detected as a background area and thus the image was initialized with this value. In the second pattern no back ground was indicated so the unfilled area is black. The small vertical mortar pieces were not located and thus are not included in the output pattern.

APPLICATIONS AND CONCLUSIONS

We have used these descriptions for recognition. In some preliminary experiments using a decision tree classifier with the symbolic descriptions, all the test samples were correctly identified. This was a test using a small number of samples (it works with large areas, not individual pixels) and a small set of texture types, but it does provide an indication of the robustness of the description process.

This work is continuing with the analysis of texture gradients including extraction of the necessary information from real images and analysis. We have applied this system to a variety of texture types and it is capable of producing effective descriptions.

REFERENCES

1. R.M. Haralick, "Statistical and Structural Approaches to Texture, Proc. IEEE 67, 1979, pp.786-804.

2. R.M. Haralick, K. Shanmugam, and I. Denstein, "Textural Features for Image Classification," IEEE Trans. SMC-6, 1973, pp. 660-621.

3. L.S. Davis, S. Johns, R.J.K. Aggarwal, "Textural Analysis Using Generalized Co-occurrence Matrices," IEEE-T-PAMI-1, 1979, pp. 251-259.

4. A. Rosenfeld and Kak, Digital Picture Processing, Academic Press: New York, 1976.

5. R. Bajcsy, "Computer Identification of Visual Surfaces," Computer Graphics and Image Processing-2, Oct. 1973, pp. 118-130.

6. J.T. Maleson, "Understanding Texture in Natural Images," University of Rochester, Ph.D. Thesis, to appear.

7. A. Rosenfeld, "Cooperative Computation in Texture Analysis," in Proc. Image Understanding Workshop, Los Angeles, Nov. 1979, pp. 52-56.

8. H. Tamura, S. Mori, T. Yamawaki, "Textural Features Corresponding to Visual Perception," IEEE Trans. SMC-8, June, 1978, pp. 460-473.

9. D. Marr, "Early Processing of Visual Information," AI-M-340, Artificial Intelligence Laboratory, MIT, Cambridge, MA. 1975.

10. F. Vilnrotter, R. Nevatia and K. Price, "Structural Description of Natural Textures," Proc. 4-ICPR, Miami Beach, Dec. 1980.

11. R. Nevatia, K. Price and F. Vilnrotter, "Describing Natural Textures," Proc. 6-IJCAI, Tokyo, Japan, Aug. 1979, pp. 642-644.

12. T. Matsuyama, K. Saburi and M. Nagao, "A Structural Description of Regularly Arranged Textures," 5-IJCPR, Miami, Fl., Dec. 1980, pp. 1115-1118.

13. L.S. Davis, "Computing the Spatial Structure of Cellular Textures," CGIP-11, pp. 111-122, 1979.

14. R. Conners, and C. Harlow, "Toward a Structural Textural Analyzer Based on Statistical Methods," CGIP-12, pp. 224-256, 1980.

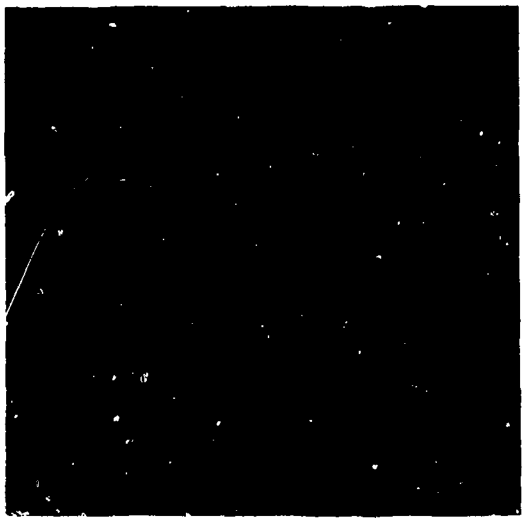


Fig. 1. Raffia subwindow.

```
NUMBERS APPEARING IN PARENTHESES ARE SCALE DEPENDENT

FILENAME = RAFFIA.NR10

DARK OBJECT DESCRIPTIONS

HORIZONTAL SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY OR PREDOMINANT
     ELEMENT SIZE

30° SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY
     WEAK EVIDENCE OF PREDOMINANT ELEMENT SIZE
        (25.40)

VERTICAL SCAN DIRECTION

     STRONG EVIDENCE OF PERIODICITY (ELEMENT
        SPACING 8.00)
     STRONG EVIDENCE OF PREDOMINANT ELEMENT
        SIZE (3.00) WITH MODERATE SUPPORT FOR
        ELEMENT SPACING (8.00)

     RATIO OF SIZE TO PERIOD IS .38


LIGHT OBJECT DESCRIPTIONS

HORIZONTAL SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY
     STRONG EVIDENCE OF PREDOMINANT ELEMENT SIZE
        (3.00)

30° SCAN DIRECTION

     NO EVIDENCE OF PERIODICITY OR PREDOMINANT
        ELEMENT SIZE

VERTICAL SCAN DIRECTION

     STRONG EVIDENCE OF PERIODICITY (ELEMENT
        SPACING 8.00)

     STRONG EVIDENCE OF PREDOMINANT ELEMENT
        SIZE (5.00) WITH MODERATE SUPPORT FOR
        ELEMENT SPACING (8.00)

     RATIO OF SIZE TO PERIOD IS .63


60°, 120° and 150° SCAN DIRECTIONS FOR BOTH LIGHT
AND DARK OBJECTS

     NO EVIDENCE OF PERIODICITY OF PREDOMINANT
        ELEMENT SIZE
```

Fig. 2. Symbolic texture description of raffif.

```
PRIMITIVE  ANALYSIS FOR TEXT. SUPP12 (THRESH = 10)

RELATIVE INTENSITY IS DARK     DIRECTION IS
VERTICAL

NUMBER OF SAMPLES: 108

     AVERAGE PRIMITIVE DIMENSIONS ARE: (2.00 AND
        10.39)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (20.30)

     AVERAGE PRIMITIVE INTENSITY IS: (128.79)

     PRIMITIVES REPEAT AT ELEMENT SPACING: (8.00)
        IN ABOVE MENTIONED DIRECTION


RELATIVE INTENSITY IS LIGHT     DIRECTION IS
VERTICAL

NUMBER OF SMAPLES: 109

     AVERAGE PRIMITIVE DIMENSIONS ARE: (4.00 AND
        9.33)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (36.94)

     AVERAGE PRIMITIVE INTENSITY IS: (172.35)

     PRIMITIVES REPEAT AT ELEMENT SPACING: (8.00)
        IN ABOVE MENTIONED DIRECTION


RELATIVE INTENSITY IS LIGHT     DIRECTION IS
HORIZONTAL

NUMBER OF SAMPLES: 68

     AVERAGE PRIMITIVE DIMENSIONS ARE: (2.00
        AND 7.88)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (15.18)

     AVERAGE PRIMITIVE INTENSITY IS: (190.47)

     NO EVIDENCE OF PERIODICITY
```

Fig. 3. Raffia primitive texture element description.

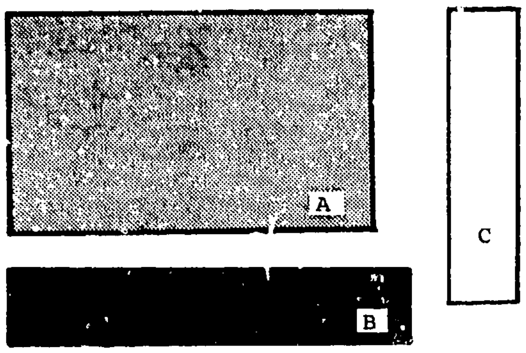


Fig. 4. Abstract representation of raffia primitives.

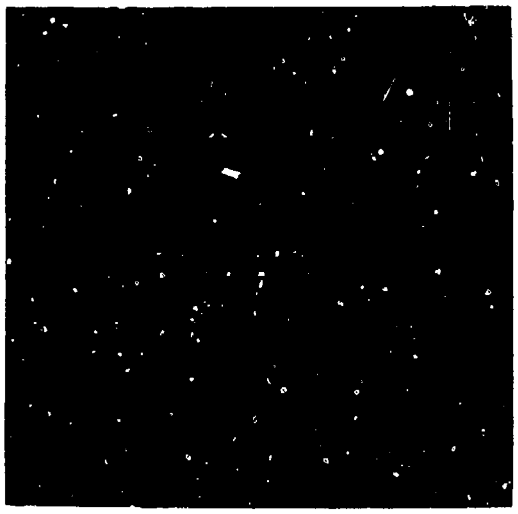
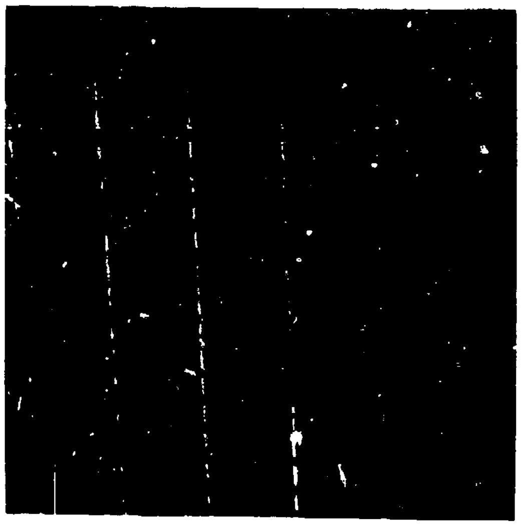


Fig. 5. Composite primitives.



(a) Brick pattern 1 subwindow.

(b) Brick pattern 2 subwindow.

Figure 6.

```
PRIMITIVE ANALYSIS FOR BRICK 1

RELATIVE INTENSITY IS LIGHT      DIRECTION IS
VERTICAL

NUMBER OF SAMPLES: 87      PRIMITIVE NUMBER IS: 1

     AVERAGE PRIMITIVE DIMENSIONS ARE: (2.00
        AND 52.92)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (105.39)

     AVERAGE PRIMITIVE INTENSITY IS: (120.80)

     PRIMITIVE REPEAT AT ELEMENT SPACING:
        (15.00) IN ABOVE MENTIONED DIRECTION

RELATIVE INTENSITY IS DARK      DIRECTION IS
VERTICAL

NUMBER OF SMAPLES: 106     PRIMITIVE NUMBER IS: 2

     AVERAGE PRIMITIVE DIMENSIONS ARE: (10.00
        AND 35.34)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (353.14)

     AVERAGE PRIMITIVE INTENSITY IS: (100.59)

     PRIMITIVES REPEAT AT ELEMENT SPACING: (15.00)
        IN ABOVE MENTIONED DIRECTION
```

Fig. 7a. Brick pattern 1 primitive texture element description.

```
PRIMITIVE ANALYSIS FOR BRICK 2

RELATIVE INTENSITY IS LIGHT      DIRECTION IS
VERTICAL

NUMBER OF SAMPLES: 39      PRIMITIVE NUMBER IS: 1

     AVERAGE PRIMITIVE DIMENSIONS ARE: (2.00
        AND 145.26)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (289.87)

     AVERAGE PRIMITIVE INTENSITY IS: (175.44)

     PRIMITIVES REPEAT AT ELEMENT SPACING: (12.00)
        IN ABOVE MENTIONED DIRECTION

RELATIVE INTENSITY IS DARK      DIRECTION IS
VERTICAL

NUMBER OF SAMPLES: 122     PRIMITIVE NUMBER IS: 2

     AVERAGE PRIMITIVE DIMENSIONS ARE: (8.00
        AND 40.93)

     AVERAGE PRIMITIVE SIZE IN PIXELS IS: (326.57)

     AVERAGE PRIMITIVE INTENSITY IS: (134.25)

     PRIMITIVES REPEAT AT ELEMENT SPACING: (12.00)
        IN ABOVE MENTIONED DIRECTION
```

Fig. 7b. Brick pattern 2 primitive texture element description.

Figure 7.

Fig. 8a. Brick pattern 1 composite primitives.

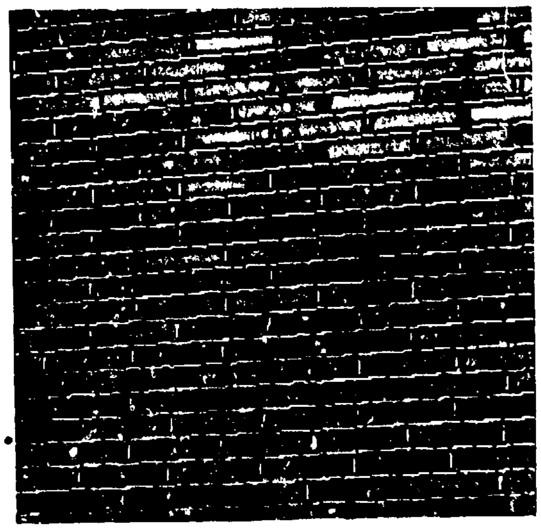


Fig. 8b. Brick pattern 2 composite primitives.

Figure 8.

BRICK1 PRIMITIVE PLACEMENT RESULTS

PRIM1	PRIM2	ANGLE[1] (DEGREES)	DISTANCE (PIXELS)	TOTAL (%)
1	1	-90	15	52.91
1	1	90	15	52.90
1	2	-90	6	***52.72
1	2	90	6	52.25
2	1	-90	6	42.88
2	1	90	6	51.48
2	2	180	45	**65.10
2	2	-90	15	*94.55
2	2	0	45	**63.12
2	2	90	15	*94.55

[1]Negative angles are counter-clockwise; positive angles are clockwise.

Fig. 9a. Brick pattern 1 interprimitive angle in degrees, distance in pixels and frequency of occurrence.

BRICK2 PRIMITIVE PLACEMENT RESULTS

PRIM1	PRIM2	ANGLE[1] (DEGREES)	DISTANCE (PIXELS)	TOTAL (%)
1	1	90	12	18.28
1	1	-90	12	18.27
1	2	-170	18	18.47
1	2	-170	30	18.88
1	2	-30	12	***28.70
1	2	10	24	23.81
1	2	10	33	18.76
1	2	80	6	20.70
2	2	-160	24	*36.63
2	2	-90	12	19.19
2	2	-30	27	**33.17
2	2	0	45	36.53
2	2	20	24	*36.41
2	2	90	12	20.86
2	2	150	27	**32.52
2	2	180	45	36.18

[1]Negative angles are counter-clockwise; positive angles are clockwise.

Fig. 9b. Brick pattern 2 interprimitive angle in degrees, distance in pixels and frequency of occurrence.

Figure 9.

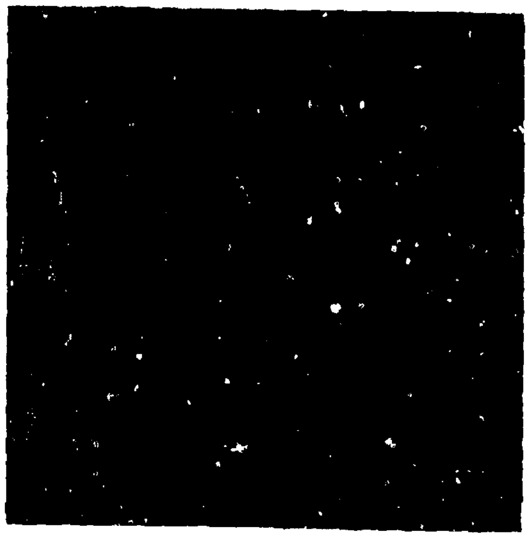


Fig. 10a. Brick pattern 1 reconstruction.

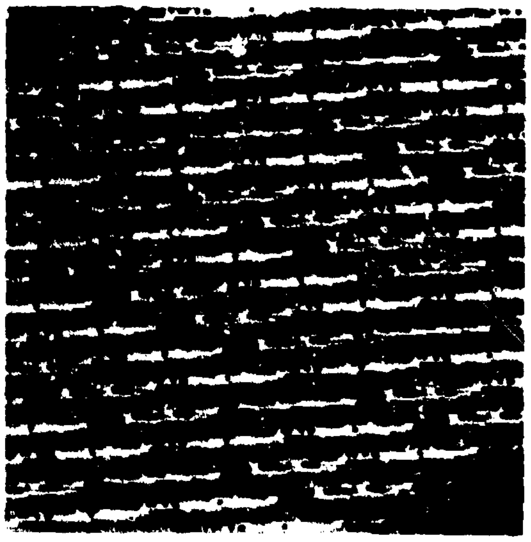


Fig. 10b. Brick pattern 2 reconstruction.

Figure 10.

COMPUTATIONAL MODELS FOR TEXTURE ANALYSIS AND SYNTHESIS

David D. Garber
Alexander A. Sawchuk

Image Processing Institute
University of Southern California
Los Angeles, California 90007

Abstract

In this paper, binary and gray-level natural textures are synthesized using several methods. The quality of the natural texture simulations depends on the computation time for data collection, computation time for generation, and storage used in each process. Many textures are adequately simulated using simple models thus providing a potentially great information compression for many applications. Other textures with macrostructure and nonstationary characteristics require more extensive computation to synthesize visually pleasing results. Although the success of texture synthesis is highly dependent on the texture itself and the modeling method chosen, general conclusions regarding the performance of various techniques are given.

1. Introduction

Texture is important characteristic for the analysis of many types of images. It is an important feature for discrimination and identification of regions in images and as a result, many techniques for texture analysis have been in the area of discrimination. Hence, many different texture discrimination techniques have been developed [13]. Most are ad hoc.

Texture synthesis has been over-shadowed by the emphasis placed on the discrimination problem and its applications. Little work has been done in the synthesis area even though numerous applications exist. For example, sensors could identify boundaries of textured image regions. Based on statistics gathered by the sensor, this region could be reconstructed using simulation techniques with little or no loss of information. The result is excellent information compression.

Texture synthesis can also be used as a texture analysis tool leading to a better understanding of textures and their perception by humans as well as improved methods of discrimination. By carefully controlling the statistics of a texture in a synthesis process visual changes in texture are observed. Thus, texture synthesis methods allow researchers to identify and measure the information content of individual statistical measurements. By assembling these measurements and incorporating them into a texture simulation process, statistics may be measured from a parent texture and used to produce a texture simulation. The degree to which the parent and simulation are visually similar indicates the value of the statistical measurements and the model used in the simulation process. Given a group of statistical measurements which are proposed to be useful texture measures, the best may be chosen based on the quality of the corresponding texture simulations. In this way, researchers are able to develop better discrimination as well as better synthesis methods.

2. Concepts of Texture Synthesis

Despite its importance, a precise definition of texture does not exist. Texture is often considered to be composed of a set of primitives and their spatial organization. More importantly, texture usually possesses an invariance property. It is this invariance property which we will use to nebulously define texture in this paper. An observer should detect no visual difference between one windowed portion of a textured region and another. Thus texture is also a function of window size. If a difference over a region is detected then either the texture is not homogeneous or a larger window should be used. Windowing is very important when gathering statistics to be used for texture discrimination or texture synthesis.

The approach to texture synthesis used in this paper is outlined in Fig. 1. As a first step in the synthesis process, statistics are calculated f[illegible]m measurements taken on a parent sample texture. The statistics are then used to estimate model parameters. In the final step, these model parameter estimates are used to generate a texture synthesis.

All of the digital images in this paper are 512 by 512 pixels. They have either 256 gray levels (continuous tone) or 2 gray levels (binary). The original parent texture images in this paper have been chosen from an album by Brodatz [1]. High quality prints obtained from the photographer were scanned and digitized at the USC Image Processing Institute.

3. Statistics of a Texture

The terminologies in a portion of previous texture work have often been vague at best. As a result, the terms second-order and third-order have been seriously twisted and misinterpreted from study to study. In this paper, we will attempt to suppress this confusion by carefully defining the various terms.

The stochastic approach toward texture analysis considers texture fields as samples of two-dimensional stochastic fields. Assuming that we are dealing with sampled and quantized imagery, let $I(n_{i1}, n_{i2})$ denote the random field. Here n_{i1} and n_{i2} are integers representing coordinates of points in the image plane. Let $\vec{n}_i$ be the vector having coordinates n_{i1} and n_{i2} (i.e. $\vec{n}_i = (n_{i1}, n_{i2})$). Second-order statistics are given by the set of second-order joint density functions

$$P_{\vec{n}_i, \vec{n}_j}(V_i, V_j) \tag{1}$$

for all possible vectors $\vec{n}_i$ and $\vec{n}_j$, where V_i and V_j are the values of the random variables $I(\vec{n}_i)$ and $I(\vec{n}_j)$, respectively. In most texture work and in all of the work in this paper (except for the work in section 13) the random field is assumed to be homogeneous, that is, all orders of probability densities are invariant through translations. Thus,

$$P_{\vec{n}_i, \vec{n}_j} \equiv P_{\vec{n}_i+\vec{c}, \vec{n}_j+\vec{c}} \tag{2}$$

where $\vec{c}$ is an arbitrary vector constant. As an example,

$$P(V_1, V_2) \equiv P(V_3, V_4) \tag{3}$$

where V_1, V_2, V_3 and V_4 are as shown in Fig. 2. In most of our work, dummy values of random variables (denoted for example by V_i) will be used to label pixels at vector location $\vec{n}_i$.

Given the assumptions that a texture field is homogeneous, the joint density functions P for all vector separations $\vec{r} = \vec{n}_i - \vec{n}_j$ represent the most complete set of second-order statistics possible. The statistical expectation of any functions of these joint density functions are called second-order statistics. If a pixel is connected to any of its neighbors on the same row, that is if we consider neighbors immediately to the left or right (such as V_5 and V_6 in Fig. 2) then their joint density is called a second-order nearest-neighbor joint density and any statistical expectations of the joint density are second-order nearest-neighbor statistics.

Similarly, third-order statistics are given by the set of third-order density functions

$$P_{\vec{n}_i, \vec{n}_j, \vec{n}_k}(V_i, V_j, V_k) \tag{4}$$

Assuming homogeneity of the texture, then

$$P_{\vec{n}_i, \vec{n}_j, \vec{n}_k} \quad P_{\vec{n}_i+\vec{c}, \vec{n}_j+\vec{c}, \vec{n}_k+\vec{c}} \tag{5}$$

for all $\vec{n}_i$ and an arbitrary vector constant $\vec{c}$. As an example,

$$P(V_1, V_2, V_3) = P(V_4, V_5, V_6) \tag{6}$$

in Fig. 3. The statistical expectations of any function of these third-order densities are called third-order statistics. All second-order statistics may be derived from third-order joint densities.

4. The Stochastic Synthesis Approach

Many early texture studies involved the use of binary textures generated by one-dimensional Markov processes. Such work was presented by Julesz [2], Pollack [3], Purks and Richards [4] and Garber [5]. In these one-dimensional models a large vector of pixels was generated line by line using a set of parameters

$$P(V_{N+1}/V_1, V_2, \ldots, V_N) \tag{7}$$

where P(A/B) represents the probability of A given B. We will refer to these conditional probabilities as generation parameters. In the above notation each V represents a generated pixel which has value 0 (black) or 1 (white). Thus each pixel value depends on the N pixels previous to it. A two-dimensional texture image is then formed by breaking up the large vector of pixels into shorter strings and stacking them one on top of the other (see Fig. 4). This procedure for large images nearly insures image row independence (unless N was large) thereby creating only horizontally oriented textures totally unsuitable for simulating natural two-dimensional textures.

By allowing N to increase exceeding the short string line length, two-dimensional (vertical and horizontal) dependence may be induced into the generating process. A pixel value then depends not only on the pixels previous to it on the same line but also on the pixels above it (see Fig. 5(b)). Thus, textures could be generated as a time sequence in television raster scan fashion. In theory, texture dependence could be extended _ad infinitum_, however practical considerations concerning the actual generation process show us that 2^N generation parameters must be accounted for. As a possible solution to the storage problem we can choose to ignore all but N of the previous pixels in our generation process and we can allow the pattern of the V_i's to become flexible. This idea will be discussed in later sections of this paper.

Throughout the remainder of this paper, the set of pixels, V_i's, on which the next pixel, V_{N+1}, depends will be referred to as the "kernel" of the synthesis process. The pixel V_{N+1} will be referred to as the "eye" of the kernel.

Given a parent texture we can estimate the generation parameters needed to generate it in many ways. Ignoring boundary conditions, linear unbiased estimates ($\hat{P}$'s) of the P's may be defined as

$$\hat{P}(V_1,V_2,\ldots,V_N) = \frac{1}{M}\sum_{j=0}^{M} \prod_{k=1}^{N} \delta(I(k+j)-V_k) \qquad (8)$$

where

$$\delta(V_j,V_k) = \begin{cases} 0 & \text{if } V_j \neq V_k \\ 1 & \text{if } V_j = V_k \end{cases}$$

and I(i) represents the ith element of the one-dimensional texture string from which the parameters are to be estimated. Equation (8) assumes that the V_i are contiguously located in order along a line. It simply states that in order to estimate $P(V_1,V_2,..,V_N)$ for a fixed pattern $V_1,V_2,\ldots,V_N$, all M substrings (samples) of length N are taken from a parent substring of length M+N-1 and the number of occurrences of the specific pattern $V_1,V_2,\ldots,V_N$ are counted, then divided by M. This is equivalent to estimating the probability density function of a random variable by the histogram of a set of samples.

This idea of estimating N-grams, $P(V_1,V_2,\ldots,V_N)$, from a sample parent texture may be extended to the two-dimensional case. A histogram of occurrences of each pattern of $(V_1,V_2,\ldots,V_N)$ is made by passing the two-dimensional kernel in Fig. 5(b) over the two-dimensional sample parent image. The tally is then divided by the total number of sample patterns observed to obtain $P(V_1,\ldots,V_N)$. As was stated earlier, two-dimensional synthesis is merely an extension of the one-dimensional case ignoring boundary conditions of the two-dimensional image.

The generation parameters of a texture may be estimated for any given set of N V 's from a parent texture using the estimates of the P's from Eq. (8). These statistics have the property

$$E[\hat{P}(V_{N+1}/V_1,\ldots,V_N)] = P(V_{N+1}/V_1,\ldots,V_N)$$

where

$$\hat{P}(V_{N+1}/V_1,\ldots,V_N) = \hat{P}(V_1,\ldots,V_N,V_{N+1})/ \qquad (9)$$

$$(\hat{P}(V_1,\ldots,V_N,0) + \hat{P}(V_1,\ldots,V_N,1)).$$

5. Kernel Selection Using The Linear Model

We will refer to the V_i's on which the next pixel, V_{N+1}, depends as the kernel of the generation process. Geometrically speaking, the V_i's form a kernel "shape" or "pattern" which may or may not be spatially contiguous. For example, in Fig. 6 a generating kernel shape is shown where the V_5 pixel directly depends on only some of the pixels in its surrounding neighborhood. In this case, V_5 may be generated based on the values of pixels V_1,V_2,V_3 and V_4 but is directly dependent on no other pixels in the neighborhood. This does not imply that V_5 is not related or correlated with its other neighbors. In fact, the relationships between V_1, V_2, V_3, V_4, and V_5 will determine other interrelationships.

A non-contiguous neighborhood of V_i's is used as it allows a more parsimonious model for texture generation to be chosen. An analogy is in simple linear regression (as defined by Draper[10]) where independent variables which do not contribute to the prediction or estimation of the dependent variable are dropped from the model equation. In texture generation this allows the model to be estimated by fewer parameters and makes the generation-synthesis process more efficient by reducing the number of computations required. When generating textures based on N-grams, reducing N reduces the amount of storage required for g^N generation parameters as defined by Eq. (9) where g is the number of gray levels in the image and N is the number of elements in the synthesis kernel. By allowing the kernel of V_i's on which V_{N+1} depends to be non-contiguous, the range of dependence in a distance sense is increased over that which would be allowed with a contiguous kernel containing the same number of V_i's. This is very important to obtain the larger structure apparent in many textures. Reducing the number of pixels in the model also relieves us from the complex numerical problems of inverting matrices of unwieldy size, a necessary step in linear model parameter estimation discussed later in this paper. We would, for example, not expect our V_{N+1} pixel to depend on a pixel V_i where the spatial separation between V_{N+1} and V_i is large. If that distance is small, however, we would expect a large dependence.

The method for choosing the proper independent variables (V_i's) to be included in the generation process requires special attention. We wish to choose the best subset of N variables from a larger finite neighborhood of T variables, where $N<T$. Evaluating such subsets and their corresponding models requires a criterion. Texture results for each possible model could be visually examined and compared and the V_i's of the model corresponding to the visually most pleasing result could be chosen. However, $\binom{T}{N}$ model evaluations must be done using this approach. For a simple search through T = 40 points with N = 12, 5.5 billion models would have to be evaluated! This approach is therefore impractical and so a sub-optimal approach which yields a good but not necessarily the best set of V_i's for our model must be used.

If we view this problem as one of predicting a dependent variable, V_{N+1} from a large set of independent variables, V_i's, then the standard linear model approaches may be applied. A forward selection procedure was used to choose the set of V_i's in the model and it is explained in detail by

Draper and Smith [10]. One at a time, the V_i's are entered into the model and the linear equation

$$V_{N+1}=\beta_1 V_1+\beta_2 V_2+\ldots+\beta_N V_N+\beta_0+\epsilon \qquad (10)$$

At each step, the V_i which minimizes the overall sum of squares error when the corresponding linear model is applied to the original input data is entered into the model. Variables are entered until either a maximum number is tained or the magnitude of the corresponding β_i coefficient is small.

5. Seeding the Process

When synthesizing a two-dimensional image, a frame of pixels is needed to seed the process as shown in Fig. 7. The seeding process may be handled in a variety of ways. The simplest approach would be to randomly generate the seed once for the whole image. In this case the pixel values in the seed frame of Fig. 7 remain the same throughout the generation process. Parent texture data could also be used to seed the generation procedure. Regardless of the seeding process, all texture synthesis methods developed in this paper normally converge to a steady state within 5 to 20 pixels of the border of the image. This was confirmed by repeated studies of convergence effects on texture simulations. In most cases, this narrow region is not noticeable and is included as a part of the result. In some critical applications these edges could be thrown away.

7. Results

Once the points for the kernel are chosen based on the linear model derived using the methods described in the previous sections, estimates of the generation parameters for the texture are obtained using concepts discussed in section 4. Practical considerations require us to use binary images (g=2) and limit N, the number of pixels in the kernel, to 12 to 18 depending on the processor storage available as 2^N values must be stored. These conditional probabilities are then used to generate each pixel along a row, row by row until a complete two-dimensional texture is obtained. For each pixel the appropriate generation parameter estimate is found and a uniformly-distributed pseudo-random variable is generated. Based on these two values, a black pixel (0) or white pixel (1) is generated.

In practice, not all of the generation parameters may be estimated when N is large because all possible patterns of $V_1,V_2,\ldots,V_N,V_{N+1}$ may not be present in the sample image or there may be few of them. Smaller samples can cause inaccurate estimation of the $P(V_{N+1}/V_1,\ldots,V_N)$ as the variance of our estimate is larger and therefore the expected error of our estimate is larger than would be expected with a larger sample size. In these cases it is important to sum over the least significant kernel elements and estimate $P(V_{N+1}/V_1,\ldots,V_N)$ by $\hat{P}(V_{N+1}/V_i,\ldots,V_N)$. In our study, this was done if the sample size to compute $\hat{P}(V_{N+1}/V_1,\ldots,V_N)$ was less than 10. The variable i is increased until this condition is met.

Texture simulations using this method are shown in Figs. 15(b), 16(b) and 17(b). Visually, the results are very good. As the estimated texture generation parameters are approximated using statistics gathered from the full parent texture, non-homogeneity in the parent texture will cause an "average" texture to be synthesized.

The bark texture is among the most difficult to simulate due to its very unusual macro-structure. Still, the N-gram simulation looks remarkably similar to the original when windowed regions 20 to 40 pixels square are observed. The parent texture of water is non-homogeneous as the waves are more closely spaced on one side that on the other. The synthesis contains waves of an average size. As we are attempting to synthesize textures and not merely "image code" the parent textures, details and non-homogeneities will be lost in the synthesis process.

8. Linear Model Generation of Binary Textures

The process of choosing the V_i's to be present in the texture generation kernel described in section 5 of this paper actually yields a simple linear model which can also be used to generate binary textures. The model which results from the determination of the generation kernel may be expressed in equation form as

$$V_{N+1,k} = \beta_1 V_{1,k}+\beta_2 V_{2,k}+\ldots+\beta_N V_{N,k}+\beta_0+\epsilon_k \qquad (11)$$

or more simply as

$$V_{N+1} = \beta_1 V_1 + \beta_2 V_2+\ldots+\beta_N V_N + \beta_0 + \epsilon \qquad (12)$$

Once the estimates of the β_i's are known, a pixel V_{N+1} may be calculated from a set of given values V_i plus an error ϵ. In one-dimensional analysis this is sometimes known as the autoregressive time series model [6]. For binary V_i a value of V_{N+1} will be produced which is non-binary. To generate binary data using this model will therefore require quantization.

In the N-gram approach to texture simulation, the randomness of the texture is induced by the generation of a uniformly-distributed pseudo-random variable during the generation process. The comparison of this value with the estimate of the generation parameter, $\hat{P}(V_{N+1}/V_1,\ldots,V_N)$, yields the next binary pixel. A similar type of randomness must occur in the generation of binary textures using the linear model of Eq. (12). This randomness is expressed in the model in the error term ϵ.

We can obtain an estimate of the distribution of ϵ in the same manner as we estimate the $\hat{\beta}$'s of the model. This may be done by applying the model to the sample data from which it was derived and observing the errors. That is, the linear model kernel is passed over the parent texture image and at each point a $\hat{V}_{N+1}$ is calculated based on Eq. (12) without the error term. Then $V_{N+1}-\hat{V}_{N+1}$ is calculated where V_{N+1} is the actual value of V_{N+1} in the parent texture. The histogram of the values can be used to estimate the distribution of ε. As one step further we could assume that has some known distribution such as Gaussian or normal, and ~ ely estimate the parameters necessary to def this distribution. In the normal distribution case, only the standard deviation (or variance) of ε needs to be estimated. The mean of ε is zero in the linear model, least-squares distribution.

Our generation process then consists of the calculation of $\Sigma\ \beta_i V_i+\beta_0$ to which we add a random, normally-distributed error term ε and this value is then quantized to 0 or 1 based on comparison with 0.5. Results using this generation method are shown in Figs. 15(c), 16(c) and 17(c). In these figures, N was allowed to be as large as 70 as only N coefficients (not 2^N) need to be stored along with $\hat{\sigma}_\varepsilon$, the estimate of the error standard deviation.

The linear model simulations are slightly inferior to the N-gram simulations but the degradation is far less than we would expect from such a massive compression of information (which is approximately 2 to 70). The results were good enough to encourage the application of the linear model to continuous-tone textures.

McCormick and Jayaramamurthy [7] were perhaps the first to make a notable attempt to simulate natural textures using this approach. Their work consisted of a discussion of the Box and Jenkins autoregressive (AR), moving average (MA) and autoregressive integrated moving average (ARIMA) models including estimation of model parameters and adequacy of model fit. A very simple model was then used to simulate two very similar textures by filling in the holes of a parent texture using the derived model. Only two textures, both exhibiting a wood-grain-like structure, were used. Similar work was done later by Tou, Kao and Chang [8]. Unfortunately, the results of their simulation of these textures were displayed using a printout of Chinese characters and so the degree of success of their method is unclear. The appearance of texture synthesis results on a computer printout will confuse most observers unaccustomed to such crude image displays. The models were again very simple and contained no more than three terms in the linear model summation. Deguchi and Morishita [9] attempted to use the linear model to segment and partition textures. Their approach was only partially successful.

In the above simulation attempts, the models used were simple. The process of collecting statistics and estimating parameters is complex. In some cases, previous authors attempted to use the complex Box and Jenkins ARIMA model which leads to difficult model parameter estimation if the number of model elements is greater than two or three.

In our study, the simpler autoregressive model is used and is allowed to contain a large number of parameters. This is possible using the assumption of homogeneity (stationarity) combined with the forward selection process of choosing non-contiguous generation kernels as described in section 5. These models are extended further by allowing second-order autoregressive models and non-stationary noise. Results of texture simulations using these models are included in this paper.

9. The Linear Autoregressive Model

In section 8 the linear autoregressive model, used to determine the elements of the generating kernel, was expressed as

$$Y_k = \vec{X}_k^T \vec{\beta} + \varepsilon_k \qquad k=1,\ldots,M \tag{13}$$

where

$$Y_k = V_{N+1,k}$$

and

$$\vec{X}_k = \begin{bmatrix} 1 \\ V_{1,k} \\ V_{2,k} \\ \cdot \\ \cdot \\ \cdot \\ V_{N,k} \end{bmatrix}$$

Here $\vec{\beta}$ is an (N+1)x1 vector of unobservable parameters and ε_k is an unobservable random variable such that $E[\varepsilon_k] = 0$. The sample number (index) is denoted by k and M is the total number of observations. We can also define the vectors $\vec{Y}$ and $\vec{\varepsilon}$ and the matrix X by

$$X = \begin{bmatrix} \vec{X}_1^T \\ \vec{X}_2^T \\ \cdot \\ \cdot \\ \cdot \\ \vec{X}_N^T \end{bmatrix} \quad \vec{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ \cdot \\ Y_M \end{bmatrix} \quad \vec{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_M \end{bmatrix} \tag{14}$$

and our model may be expressed as

$$\vec{Y} = X\vec{\beta} + \vec{\varepsilon} \tag{15}$$

In equation form, dropping the k subscript, the model becomes

$$V_{N+1} = \beta_1 V_1+\beta_2 V_2+\ldots+\beta_N V_N+\beta_0+\varepsilon \tag{16}$$

Sums and sums of squares leading to the calculation of the correlation or covariance matrix of the parent texture are obtained by passing any chosen generation kernel pattern over the texture. From this matrix, the least-squares parameter estimate of β is obtained. The multiple-pass forward selection process described in section 5 leads to a final linear autoregressive model which is then used to generate textures.

The linear (autoregressive) model of Eq. (16) was used to simulate a variety of natural textures. Stationary, independent Gaussian noise was used to drive the synthesis process. The variance of the noise was estimated by applying the model to the sample data and observing the prediction errors. These errors, which are often called residuals, are pixels formed by the difference $V_{N+1}-\hat{V}_{N+1}$ where V_{N+1} is the actual observed pixel value of the sample parent image and $\hat{V}_{N+1}$ is the corresponding fitted value obtained by use of the linear model. The standard deviation of these errors can be measured and used as the standard deviation of pseudo-random normally-distributed noise in the generation process. Actually, this information can also be obtained during the decomposition of the covariance matrix.

To tie this method with the previous sections, this method of texture synthesis is equivalent to defining $P(V_{N+1}/V_1,\ldots,V_N)$ as a normal distribution with mean

$$\beta_1 V_1+\beta_2 V_2+\ldots+\beta_N V_N+\beta_0 \tag{17}$$

and variance VAR[ε].

The number of pixels in each generation kernel, N, varied from 30 to 60. The simulation results are shown in Figs. 18(b), 19(b) and 20(b).

These simulations indicate that the linear model using stationary gaussian noise produces acceptable simulations of a variety of textures including wool and water.

10. Second-Order Linear Model

When we say that a model is a linear or nonlinear, we are referring to linearity or nonlinearity in the parameters. The value of the highest power of an independent variable in the model is called the order of the model. For example,

$$Y = \beta_1 V_1+\beta_{11} V_1^2+\beta_0+\varepsilon \tag{18}$$

is a second-order linear model. A general second-order linear model with two independent variables may be written as

$$Y = \beta_1 V_1+\beta_2 V_2+\beta_{11} V_1^2+\beta_{22} V_2^2+\beta_{12} V_1 V_2+\beta_0+\varepsilon \tag{19}$$

A full second-order model with N independent variables will employ $(N^2+3N)/2$ terms in addition to the β_0 (constant) and ε (error) terms. This general second-order linear model may be written as

$$\begin{aligned} V_{N+1} &= \beta_1 V_1+ {}_2V_2+\ldots+ {}_N V_N \\ &\quad +\beta_{12} V_1 V_2+\ldots+\beta_N V_N^2+ \beta_0+\varepsilon \\ &= \sum_{i=1}^{N} \beta_i V_i+ \sum_{i=1}^{N}\sum_{j=1}^{N} \beta_{ij} V_i V_j+ \beta_0+\varepsilon \end{aligned} \tag{20}$$

Second-order models have been particularly useful in studies where surfaces must be approximated by polynomials of low order. In all cases, a second-order model will "fit" given data as well as or better than a first-order model that is a subset of second-order models. This does not imply that the second-order model will be more correct however, as the process which we are attempting to model may be in fact a linear first-order process or some other type.

The use of a second-order model to approximate the surface of the general stochastic model could have many advantages over a first-order model. An example of fitting such a model in one dimension to a given set of data is shown in Fig. 8.

Still the linear first-order model may provide a good fit to the data and the magnitude of the unexplained variance in the data may be large enough that the improvement due to the addition of second-order terms to the model may be barely noticeable. In two dimensions, the fitting problem is one utilizing a quadric surface such as a elliptic paraboloid or hyperbolic paraboloid versus a plane to fit a given set of data. Again, the fit may or may not be markedly better. Adding second-order terms to a model will always produce a fit as good as or better better than a first-order model but the number of computations required to compute the coefficients and fit the model are much greater.

It is also important to note that the covariances of the V_i are required in order to obtain least-square estimates of the parameters β_i in the first-order model [10]. Covariance is essentially a second-order statistic. Therefore, estimating the parameters of a second-order model will require the use of fourth-order statistics. Specifically the correlation of terms $V_{i1}V_{i2}$ and $V_{i3}V_{i4}$ is needed. This may cause serious problems as many cases the variables in a second-order model will be highly intercorrelated. For example, the terms V_1, V_1^2 and V_1V_i (if V_i is highly related to V_1) may be strongly correlated. This situation, often referred to as multi-collinearity, may cause problems during the inversion or decomposition of the estimated correlation matrix, a necessary step in model parameter estimation. For this reason, care should be exercised during the analysis of second-order models.

Inside a circular radius of 14 pixels from V_{N+1} there are 307 pixels. To search all possible cross products in this region to find the most significant would require over 47,000 cross products to be examined. Computation of a covariance matrix containing all of these terms is impossible (in practice). In our study we were limited to investigate only 820 possible cross products for entry into the generation model. As most of the variance was explained by the linear terms of the model, most of the cross products were insignificant from a statistical point of view. This selection procedure is detailed in [10] and in section 5. Those that were significant were entered into the model and a new texture was generated using Eq. (20) with stationary Gaussian noise and having zero mean and fixed variance .

The results of texture simulations using the second-order linear model are shown in Figs. 18(c), 19(c) and 20(c). On some of these textures only a slight improvement from the addition of second-order terms may be seen. In most cases, no change can be observed even when the results are displayed on a high-resolution display device. The lack of improvement could be due to the small number of cross-terms examined; however we feel that this number is sufficiently large to show any considerable improvement due to the addition of second-order terms to the linear model.

11. Textures with Non-Stationary Noise

Applying a texture generation model to the original parent texture image data used to estimate its parameters gives a residual error image. This sequence plot of residuals in the case of two-dimensional texture synthesis is essentially an image containing the pixel differences or residuals $V_{N+1}-\hat{V}_{N+1}$. Here $\hat{V}_{N+1}$ is the prediction of the next pixel in the sequence as a linear function of the pixel around it according to the model without any noise added. Naturally, we would expect these errors to be small as merely subtracting one pixel from its nearest neighbor would yield a small value in most natural, low-noise images. Definite patterns are seen to exist in these images and thus a violation of the independent assumption is indicated. Ideally, this residual image would be uncorrelated noise.

The distribution of this error and the relationships between the predicted and actual pixel values was utilized to generate textures using non-stationary noise. The procedure begins by generating a pixel V_{N+1} according to Eq. (20) excluding the error term. With this predicted value a random error value is chosen to be added to $\hat{V}_{N+1}$. This error value is chosen from the distribution of error as a function of $\hat{V}_{N+1}$ and can have any arbitrary distribution. The next pixel will than be computed in a similar manner. Results of texture synthesis formed using this model are shown in Figs. 18(d), 19(d) and 20(d).

The arbitrary distribution of error as a function of V_{N+1} is calculated by applying the calculated linear model to the original parent texture and computing a histogram of errors as a function of V_{N+1}. In other words, the distribution of ε depends on the value of $\hat{V}_{N+1}$ and this distribution is estimated by applying the model to the original parent texture and observing V_{N+1} and the error $V_{N+1}-\hat{V}_{N+1}$.

In most cases, considerable improvement is seen when these simulations are critically observed on a high-resolution display device and compared with the stationary model results. Of course, the information required to generate them is considerably greater also. The distribution of errors as a function of $\hat{V}_{N+1}$ must be condensed and coded to some degree to minimize storage requirements. For a 256-grey-level image $\hat{V}_{N+1}$ usually ranges from -50 to 305 and the errors, $\hat{V}_{N+1}-V_{N+1}$, from -255 to +255. These ranges were determined experimentally. This would yield quite a large amount of data if fully stored. By storing a small number (under 100), typical errors for each range (and not each single value) of $\hat{V}_{N+1}$ the number of data values we are required to store can possibly be reduced to under 1000. Therefore it is believed that this approach of using non-stationary, non-Gaussian noise to generate textures may be quite acceptable even with severe storage limitations.

12. Skip-Generate Method

Simulating textures which have a fine structure is usually a much easier process than simulating textures with coarse structure. This occurs because the linear model contains fewer terms if the texture pixels become uncorrelated over a small distance. For the same texture at a greater magnification, the pixels become highly correlated and the linear model will be forced to contain more terms. As the texture becomes more coarse, more time-consuming statistical measurements must be taken on the parent texture over larger windows. Motivated by these problems, the texture generation algorithms in this section have been developed.

In the texture work so far, pixel V_{N+1} was generated based on pixels above or to the left of it (see Fig. 5(b)). As discussed in section 5, the kernel does not have to be contiguous. This kernel shape is chosen to insure that the image space of our synthesized texture was filled during the generation process. However, generating pixels along a row, row by row is not the only way of filling an image space.

Consider the non-contiguous kernel mask in Fig. 9. If the spacing between the pixels in this mask is 8, using the linear model in Eq. (16) to generate the right-most pixel in the bottom row, we can generate every 8th pixel along every 8th row. At each step the next pixel is generated based on the previously-generated pixels around it (ignoring boundary conditions). After generating an image with this type of spacing, the pixels midway between the previously-generated pixels on each row may be generated using the mask in Fig. 10. In this mask, the pixel with the "x" in it denotes the next pixel, V_{N+1}, to be generated according to Eq. (17). Finally, the linear model used in this step will have different coefficients than the previous one. It is also interesting to note that new pixels depend not only on previously generated pixels above them (as with the mask in Fig. 5(b)) but depend also on the pixels below them. Still, ignoring boundary conditions, each pixel depends only on previously generated pixels. At the next step a mask similar to that in Fig. 11 can be used to fill in the pixels midway between the previously-generated pixels in each column. Again pixels are allowed to depend on pixels around them.

By repeatedly using the masks in Fig. 10 and Fig. 11 with successively closer and closer pixel spacing, the texture simulation image space is filled. An example showing the pixels generated at each successive pass is shown in Fig. 12. More importantly, to determine the linear model for each mask, only one covariance matrix is required and can contain as many or as few terms as desired. The process of collecting statistics for one matrix is not beyond the complexity that we would want to undertake for the small number of times required by this process. Naturally, any other stochastic process may be substituted for the linear model. As before, only the measurements required to estimate the parameters corresponding to each mask need to be taken. This number depends on the spacing of the pixels in the first mask, which should be a power of two. Other odd-shaped kernels and kernels whose spacing is not a power of two could be designed to form space-filling sets. Most would require more models to be estimated and would provide little additional information.

Texture simulations using this method are shown in Figs. 18(e), 19(e) and 20(e). Only a slight improvement is seen in some of the texture simulations over the synthesis done by the earlier single linear model. Most of these textures are apparently well simulated by a carefully chosen model and the results are not critically dependent on the coarseness of the textures.

A word of caution should be added concerning the computations involved in the linear model coefficient calculation of this method. During the later stages of the skip-generate method, the pixels in the generation kernel become highly correlated as the distance between them decreases with each pass. This may cause the correlation or covariance matrix of the model to be ill-conditioned. To avoid numerical problems, the number of variables entered into the process, and therefore the number of steps involved in the matrix decomposition process, should be kept to a minimum in some cases. The use of ridge regression techniques [11] might also be considered.

13. Best-Fit Texture Model

A method of generating texture simulations according to their Nth order densities was investigated for binary textures in section 4. The simulations resulting from this Markov process resembled their parental textures quite closely in most cases. When applying a similar concept to multi-grey level imagery, the limits of computer storage are soon reached. To circumvent this constraint, a new method of texture synthesis was developed and applied to a number of textures.

In binary texture generation based on N-grams a single functional value $P(V_{N+1}/V_1,\dots,V_N)$ was stored for each possible pattern $(V_1,V_2,\dots,V_N)$ where the V_i's can be zero or one. This value, also called a generation parameter, represented the conditional probability that the next pixel, V_{N+1}, in the generation process would be a zero-valued, black pixel. The V_i's were chosen by a best linear model fit detailed in section 5 and therefore the kernel of previous pixels $(V_1,\dots,V_N)$ is not necessarily contiguous (see Fig. 6). Details concerning the estimation of $P(V_{N+1}/V_1,\dots,V_N)$ from a parent texture are given in section 4. For binary textures, this single value is sufficient to define the distribution of V_{N+1} given $V_1,\dots,V_N$. The number of different functions which must be stored is 2^N. In the generation process each pixel V_{N+1} is generated based on the values of the pixels $V_1,\dots,V_N$ surrounding it and on a computer-generated uniformly-distributed random variable. The texture simulations are generated pixel by pixel along a row until each row is complete. Pixel generation along the edges of an image can be handled in a variety of ways but in section 4 pixels in these border regions were assumed to be any random value, 0 or 1, if they were outside the image boundaries.

A similar approach could be used to generate multi-grey-level textures. For a texture containing g grey levels, g^{N+1} different functions, $P(V_{N+1}/V_1,\dots,V_N)$, must be stored. (Actually only $(g-1)\cdot g^N$ are required as $\sum_{X=0}^{g-1} P(X/V_1,\dots,V_N)=1$ for all V_i). Storage limitations are soon reached. Also estimation of $P(V_{N+1}/V_1,\dots,V_N)$ is difficult as multiple occurrences of the pixel pattern $V_1,\dots,V_N$ may not exist in the parent texture. Therefore even without storage limitations the problems of estimating $P(V_{N+1}/V_1,\dots,V_N)$ from a given parent texture, which represents the distribution of V_{N+1} given the values of $V_1,\dots,V_N$ is complex.

This estimation problem no doubt has a number of _ad hoc_ solutions. The problem is basically that for large N and/or large g, there may not be a suitable number of occurrences of the pattern $V_1,\dots,V_N$ to adequately estimate the distribution $P(V_{N+1}/V_1,\dots,V_N)$ given a finite sample size. Even though a certain pattern never occurs or rarely occurs in our sample parent texture it is not implied that such a pattern is impossible and will never occur in our simulation synthesis. We might often find numerous occurrences of this pattern if our sample size or the size of our parent texture was increased, especially in noisy and fine-structured textures. But as this very large sample may not be present, we must estimate $P(V_{N+1}/V_1,\dots,V_N)$ for all $V_1,\dots,V_N$ based on available samples.

One approach would be to use sample patterns which closely resemble but which may not be exactly the same as each pattern $(V_1,\dots,V_N)$. That is in a pictorial sense, we use patterns of $(V_1,\dots,V_N)$ which look "close to" the pattern for which we are attempting to estimate $P(V_{N+1}/V_1,\dots,V_N)$. Therefore samples in our sample parent texture may be used to estimate numerous $P(V_{N+1}/V_1,\dots,V_N)$ and not just those they fit exactly. The concept of a distance function must be used to numerically define "close to". Given two patterns, one from our sample texture and the other from the conditional probability of the kernel we are attempting to estimate, the distance measure can be used to determine the value of that sample in estimating $P(V_{N+1}/V_1,\dots,V_N)$. If the fit between the kernel pattern and the pattern in the sample texture is good the associated value of V_{N+1} in the parent texture will be valuable in estimating $P(V_{N+1}/V_1,\dots,V_N)$.

Normally, when N and g are small or when we have many samples for any given $V_1,\dots,V_N$, we can use the histogram of the associated V_{N+1} to estimate $P(V_{N+1}/V_1,\dots,V_N)$. Here the relative number of times a particular value of V_{N+1} occurs given a pattern indicates the conditional probability we are attempting to estimate. This was discussed in section 4. Where a distance measure is used instead, a good fit could be considered to be synonymous with high frequency of occurrence of that pattern and a poor fit with low frequency of occurrence.

If such a method of estimating these conditional probabilities is used we are still faced with a huge storage problem. For this method to be practical, the storage requirement must be reduced. From an information standpoint, it is interesting to note that a method of estimating N-grams or conditional probabilities $P(V_{N+1}/V_1,\dots,V_N)$ from a sample parent texture image produces g^{N+1} data values from M pixel samples where M is the size of the square parent texture image in pixels. For large g and N this is a drastic increase in data. But the actual information content can really never be greater than that content of the sample parent texture image. Therefore, this M value represents an upper bound on the amount of data we should use to generate a texture simulation. Any amount of data exceeding this will contain redundant, useless data.

Combining this concept of upper bound with the idea of forming a distance measure to compare two texture kernel patterns leads to a new texture synthesis method. In this method, we generate the next pixel based on the pixels in the kernel surrounding it (see Fig. 5(b)) and their comparison to similar kernels in the parent texture. This comparison is made by passing the kernel currently present in the simulation process over the parent texture and computing the distance function at all possible points (see Fig. 13). Denoting the pixels in the parent texture by $X_{i,j}$, $i,j=0,\dots,\sqrt{M}-1$ and the pixels in the kernel $V_1,\dots,V_N$ by $Y_{i,j}$, we can compute a comparison image

$$WMSD_{a,b} = \sum_i \sum_j (X_{i+a,j+b} - Y_{i,j})^2 \cdot W_{i,j} \tag{21}$$

where

$$W_{i,j} = \frac{1}{(i-i_{NEXT})^2 + (j-j_{NEXT})^2} = \frac{1}{R^2} \tag{22}$$

where R is the euclidean distance between pixel $Y_{i,j}$ and the kernel eye Y_{NEXT} and the coordinates of the eye are given by (i_{NEXT}, j_{NEXT}).

As the first step in comparing a given kernel $Y_{i,j}$ to all kernels in the parent texture, for each point (a,b) in the parent texture, ignoring edges, the WMSD is computed resulting in an image of WMSD's. Where the fit between the generated kernel $Y_{i,j}$ and the image $X_{i,j}$ is good, we would expect $WMSD_{a,b}$ to be small. The smallest WMSD represents the "best" fit according to our norm. We could choose the Y_{NEXT} associated with this best fit at point (a,b) to be our next pixel in the generation process, however this can cause problems. First of all, the generation process would "lock in" on the parent texture and the generated texture could very well become just an exact copy of the input parent texture. Second, we know ideally that Y_{NEXT} has a distribution, not just a mean. In the autoregressive model of section 9 we gave Y_{NEXT} a distribution by adding random noise to it. Although this could be done here, such an approach would fail to use additional information contained in the WMSD image. There may be a set of points (a,b), all exhibiting a good fit to the kernel pattern $Y_{i,j}$. In fact, the best fit may have a noisy Y_{NEXT} and the other good fits could provide information to improve the prediction of the Y_{NEXT} in the generation process. Using a set of best fits is equivalent to increasing our sample size. We look at a set of similar patterns to pick our Y_{NEXT}.

At this point there are numerous ways to proceed. Logically those patterns with the "best" fit should provide better estimators for Y_{NEXT} so some kind of weighting decision is needed to choose the relative importance of the WMSD's found. If we search through the WMSD image and find the minimum value, $WMSD_{min}$, and scale all the WMSD's by that we form a new image MAX1

$$MAX1_{a,b} = \frac{WMSD_{min}}{WMSD_{a,b}} \tag{23}$$

This image has the value 1.0 at the best fit point and values $0 \le MAX1 \le 1.0$ elsewhere.

Here we can look at the MAX1(a,b) image and study its range. If $0.16 \leq \mathrm{MAX1} \leq 1.0$ it is implied that the worst fit yields a $\overline{0.16}$ value. Somehow that worst fit should be translated to imply that the probability of choosing the Y_{NEXT} associated with that point $(a,b)_{worst}$ is nearly 0.0. The simplest way of doing that is to take powers of the image MAX1(a,b). The maximum remains 1.0 while smaller numbers approach 0.0. For example $(1.0)^{10}=1.0$ but $(0.16)^{10}=1 \times 10^{-8}$. We do this to obtain an ad hoc estimate of $P(Y_{NEXT}/Y_{i,j})$. After experimentally studying the values of MAX1(a,b) and its powers, the value of 16 was chosen and a new image PDFUNS

$$\mathrm{PDFUNS}_{a,b} = (\mathrm{MAX1}_{a,b})^{16} \tag{24}$$

was used to estimate the probability density function $P(Y_{NEXT}/Y_{i,j})$. PDFUNS is then scaled so that $\sum \mathrm{PDFUNS}(a,b)=1$. Finally a uniformly distributed random variable, r, [0,1] is generated and a point (c,d) is found such that

$$\sum_{a=1}^{c-1}\sum_{b} \mathrm{PDFUNS}_{a,b} + \sum_{b=1}^{d-1} \mathrm{PDFUNS}_{c,b} < r$$

$$\sum_{a=1}^{c}\sum_{b} \mathrm{PDFUNS}_{a,b} + \sum_{b=1}^{d} \mathrm{PDFUNS}_{c,b} < r \tag{25}$$

The Y_{NEXT} associated with the kernel shape at (c,d) is then used as the next pixel in the generated image. The process is continued until a full texture image is generated with the kernel window moving one pixel at each step, row by row.

In an indirect way, this is equivalent to generating a random variable having any distribution using the desired cumulative distribution combined with a uniformly distributed random variable (which is easy to generate). In other words, uniformly-distributed deviates are transformed to deviates having the desired distribution using the inverse cumulative density function. This is frequently done in simulations.

For a kernel containing 55 pixels (see Fig. 14) passed over a 128x128 parent texture approximately 7.2×10^6 operations (additions or subtractions) are needed to get the WMSD image defined by Eq. (21). Another 2.6×10^5 are required to find the next pixel according to Eq. (25). therefore, to generate a 512x512 texture requires 1.96×10^{12} (2 trillion) operations.

Results from texture synthesis done by this method are shown in Figures 18(f), 19(f) and 20(f). Each of these images is 512x512 pixels. A 128x128 section of each original (parent) texture was used for the simulation. Bark exhibits very large macro structure and this is lost in the simulation. Still this simulation is better in many ways than those obtained using other models. The large number of operations makes this process very time consuming even when a pipelined processor is dedicated to the task. About 5.5 days of dedicated time on an AP120B were required to generate each texture.

Although this method is of little practical use due to the computational complexity of the algorithm a few points should be made. With constantly increasing computer processing speeds, a simplified version of this texture simulation method may be implemented in the near future. It is even possible that such computations could be performed by an array of microprocessors. In any case such brute-force approaches are simple and could be made cost-effective in the future. The results also indicate visually the amount of texture information present in a 55 pixel window (see Fig. 14) because at each pixel generation step, the next pixel in the Markov process depends on only the pixels in this neighborhood.

Finally, this approach is admittedly ad hoc. Numerous distance measures could replace the one chosen in this work and each would give different results that might appear better or worse. It is always important that the process be random and not merely copy the texture sample. If the simulation region is much larger than the parent sample, a deterministic process will quickly generate patterns that can easily be seen to repeat. In other words, the histogram represented by $P(V_{N+1}/V_1,\ldots,V_N)$ should rarely be a delta function. A reduction in the number of computations could be made if the kernel was non-contiguous. Also, better results could probably also be obtained if the kernel window was larger. The shape, contiguity and size of the kernel in this study was chosen primarily for computer programming considerations.

The results from this best-fit texture synthesis method are very pleasing but the number of computations required is large. Other similar algorithms could be developed which are simpler and could possibly produce even better results. With the decrease in computation costs and the increase in processor speeds of future computers, such texture synthesis methods could be easily implemented in the future.

14. Conclusions

The N-gram model of section 4 was an extension of earlier one-dimensional studies applied to two-dimensional natural textures. The results were very good even with the severe constraint imposed by the upper limit on the number of pixels allowed in the generation kernel. In section 8, the binary linear model, which was used to determine the contents (shape) of the generation kernel (see section 5), was used to generate binary textures. The textures generated using this model were nearly equal in quality to those of the more complex and storage-consuming N-gram model. The N-gram model of section 4 uses a generation kernel whose contents (shape) depends on the linear model. Therefore, the number of computations required in the statistics collection portion of the N-gram model necessarily includes computations of the linear model. However, in some cases, points which lie far from the kernel eye can be neglected in the N-gram model as only the best few are used due to storage limitations. On the other hand, such points should be included in the linear model therefore a larger neighborhood surrounding the kernel eye should be used in the estimation of the linear model.

In section 9, the linear autoregressive model was applied to 256-gray-level imagery. The results were good considering the vast reduction of information caused by the statistics collection process. Slightly better results were obtained by allowing the model to contain cross terms but the resulting complexity suggests that the change in texture quality is not worth the added effort and computational expense. Using non-gaussian, non-stationary noise in the model produced slightly better results but with a requirement of slightly increased storage.

The skip-generate method of section 12 may be used to improve the simulation of textures having a coarse structure. The model produces results equal in quality to the linear autoregressive model while requiring fewer computations in the collection process.

The best-fit model of section 13 represents a brute-force approach to texture synthesis. Though computationally demanding, the final results show that excellent texture simulations can be generated using complete statistics from a relatively small neighborhood. The problems with a small neighborhood are seen in the simulation of textures such as bark where the size of the primitives in the texture is much greater than the window used in the best-fit calculation.

It would be unwise to believe that all textures could be generated using any single approach, especially one which promises to compress texture information to a handful of numbers. Yet this is precisely what has been attempted in the texture synthesis work of this paper. These method could be modified and combined to form even more powerful texture synthesis techniques.

It is important to note the power and complexity of each synthesis method of this paper. Many textures can be simulated well using simple models such as the autoregressive model if the model is carefully constructed. Improvements in texture simulation were made by modifying these models and allowing them to become more complex and use more information in the generation process. Other textures require more complex models such as the best-fit model. The shortcomings of each method will constantly indicate where future work can be done.

15. References

[1] P. Brodatz, Textures: A Photographic Album for Artists and Designers. New York: Dover, 1966.

[2] B. Julesz, "Visual Pattern Discrimination," IRE Trans. on Info. Theory, vol. IT-8, pp. 84-92, Feb. 1962.

[3] I. Pollack, "Discrimination of Third-Order Markov Constraints within Visual Displays," Perception and Psychophysics, vol. 13, no. 2, pp. 276-280, 1973.

[4] S.R. Purks and W. Richards, "Visual Texture Discrimination Using Random Dot Patterns," J. Opt. Soc. Am., vol. 67, pp. 765-771, June 1977.

[5] D.D. Garber, "One-Dimensional Texture Pattern Generation and Discrimination," USCIPI Report #840, Image Processing Inst., Univ. of Southern California, Sept. 1978.

[6] G.E.P. Box and G.M. Jenkins, Time Series Analysis: Forecasting and Control, San Francisco: Holden-Day, 1976.

[7] B.H. McCormick and S.N. Jayaramamurthy, "Time Series Models for Texture Synthesis," Int. J. of Computer and Info. Sciences, vol. 3, pp. 329-343, Dec. 1974.

[8] J.T. Tou, D.B. Kao, and Y.S. Chang, "Pictorial Texture Analysis and Synthesis," Proc. of Third Int. Joint Conf. on Pattern Recognition, Coronado, Calif., Nov. 1976, p. 590.

[9] K. Deguchi and I. Morishita, "Texture Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques," IEEE Trans. on Computers, vol. C-27, pp. 739-745, Aug. 1978.

[10] N. Draper and H. Smith, Applied Regression Analysis, New York: John Wiley & Sons, 1966.

[11] J.R. Alldredge and N.S. Gilb, "Ridge Regression: An Annotated Bibliography," Int. Statist. Rev., vol. 44, no. 355, 1976.

[12] D.D. Garber, USCIPI Report 1000, Image Processing Institute, University of Southern California, Los Angeles, to be published May 1981.

[13] R.M. Haralick, "Statistical and Structural Approaches to Texture," Proc. of the IEEE, vol. 67, no. 5, May 1979.

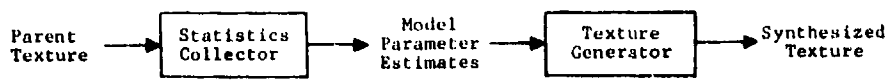


Fig. 1. Approach to texture synthetis.

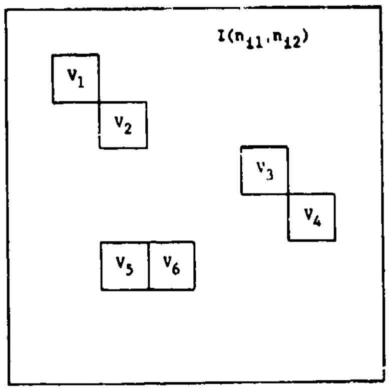


Fig. 2. Second-order statistics.

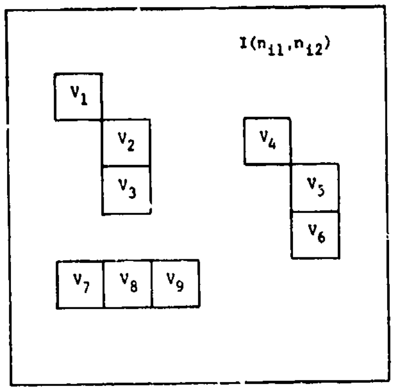


Fig. 3. Third-order statistics.

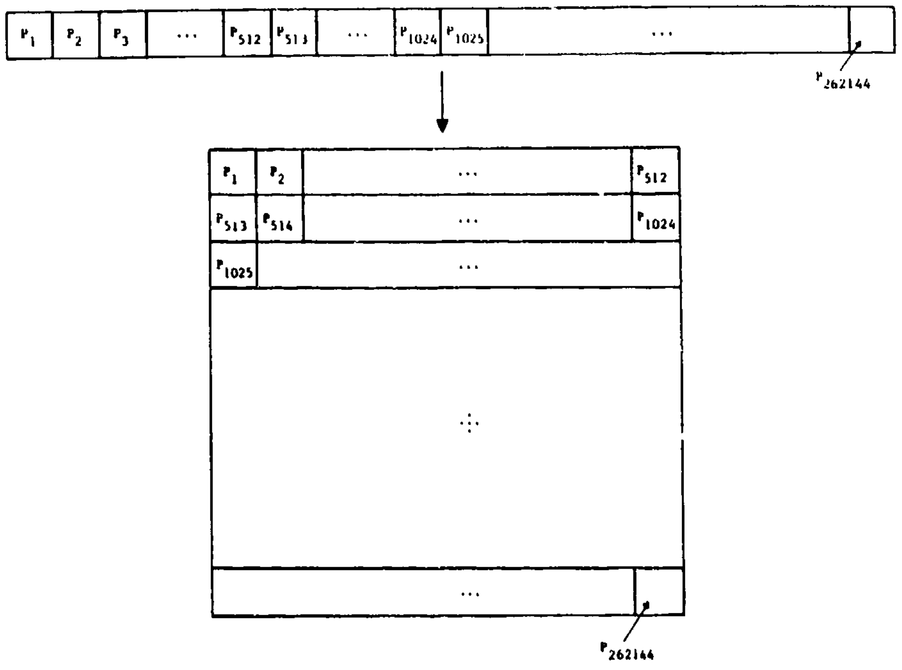
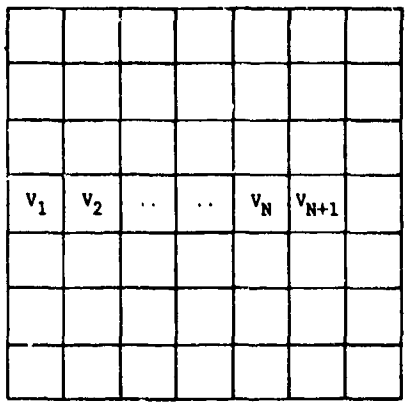
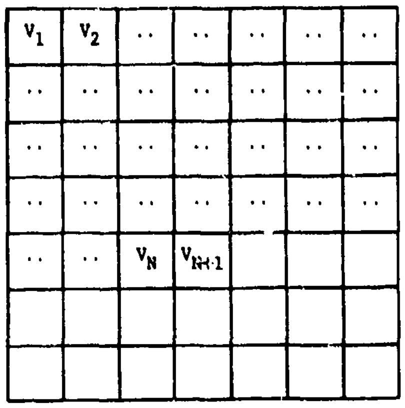


Fig. 4. Forming a two-dimensional image from a one-dimensional sequence.



(a) One-dimensional texture synthesis kernel.



(b) two-dimensional texture synthesis kernel.

Fig. 5.

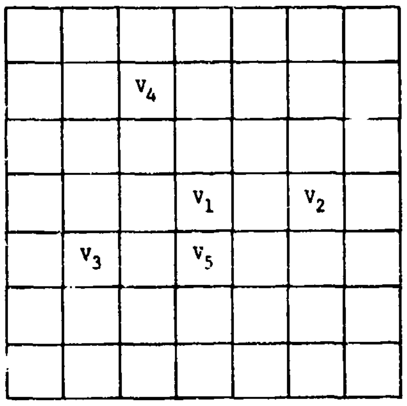


Fig. 6. Two-dimensional non-contiguous texture generation kernel.

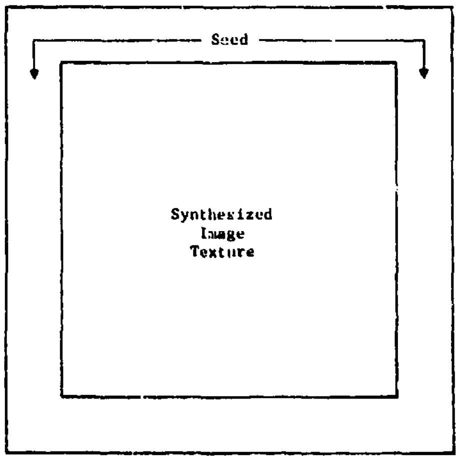


Fig. 7. Synthesized image texture and seed region.

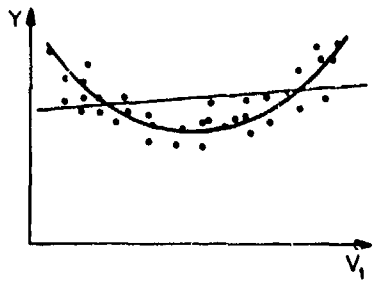


Fig. 8. Second-order linear model.

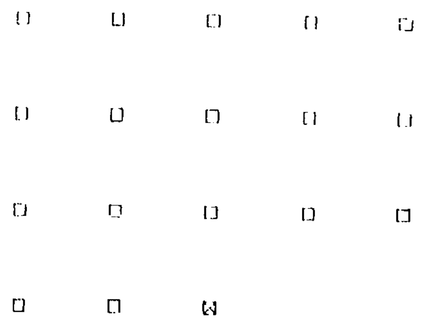


Fig. 9. First-pass generation kernel.

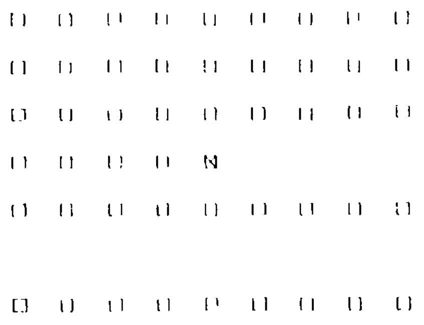


Fig. 10. Second-pass generation kernel.

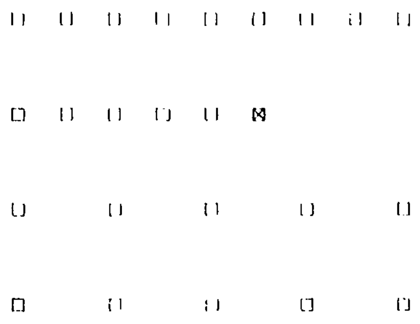


Fig. 11. Third-pass generation kernel.

1	6	4	6	2	6	4	6	1	6	4	6	2	6	4	6	1
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
3	6	4	6	3	6	4	6	3	6	4	6	3	6	4	6	3
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
1	6	4	6	2	6	4	6	1	6	4	6	2	6	4	6	1
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
3	6	4	6	3	6	4	6	3	6	4	6	3	6	4	6	3
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
1	6	4	6	2	6	4	6	1	6	4	6	2	6	4	6	1

Fig. 12. Filled space of skip-generate kernel.

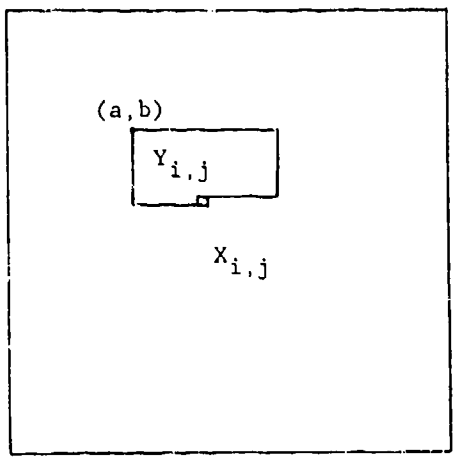


Fig. 13. Passing kernel over parent texture.

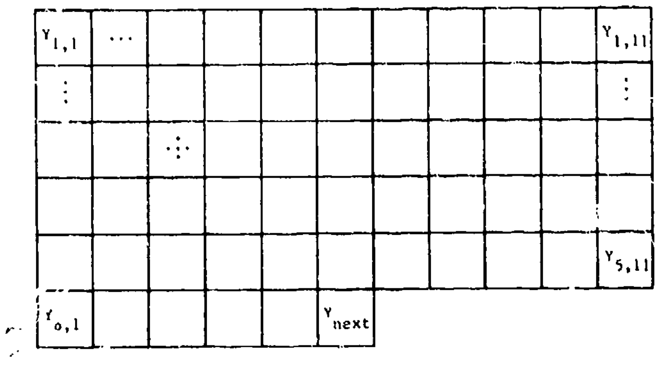
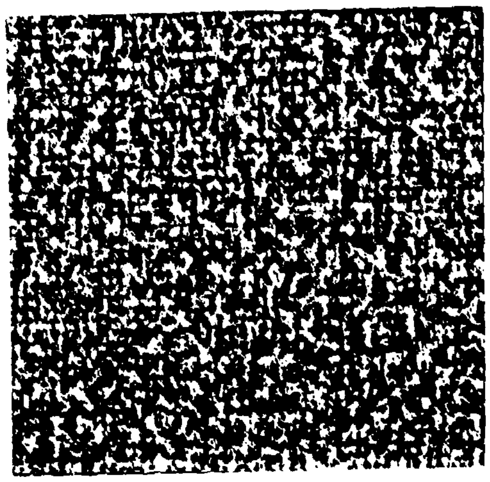
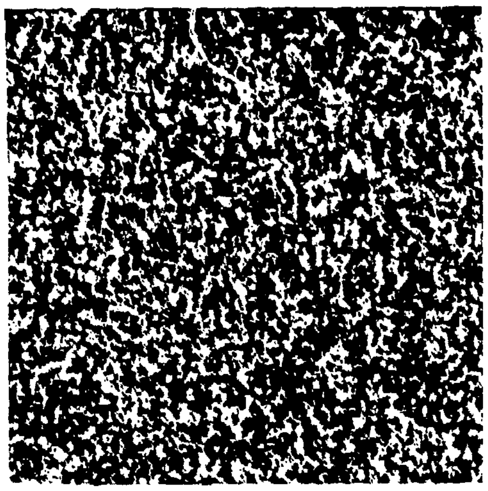


Fig. 14. Best-fit model kernel.


(a) Original


(b) N-gram

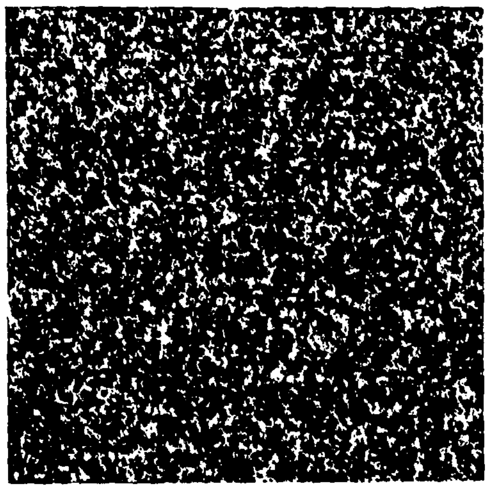
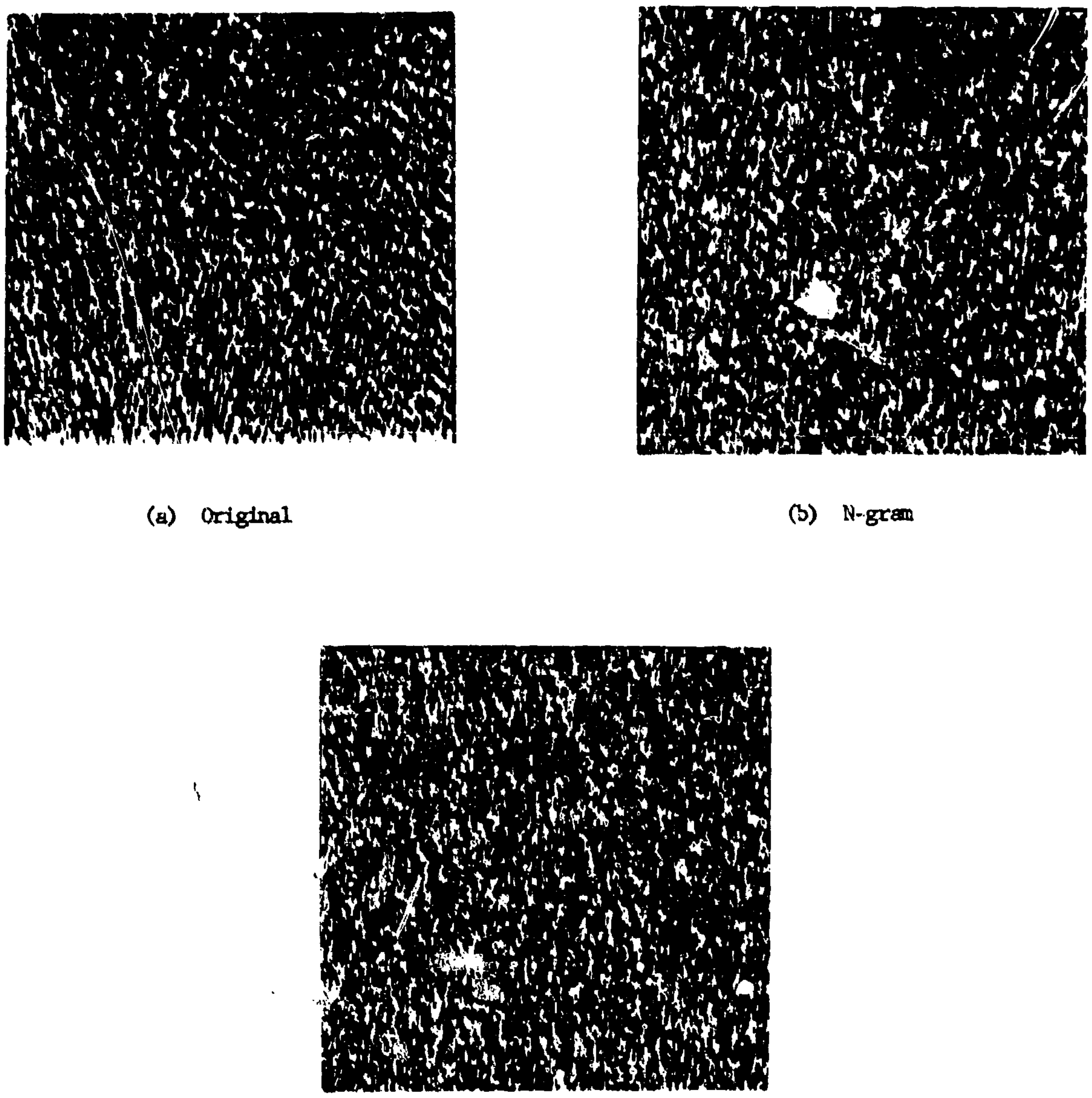

(c) Linear model

Fig. 15. Binary wool.

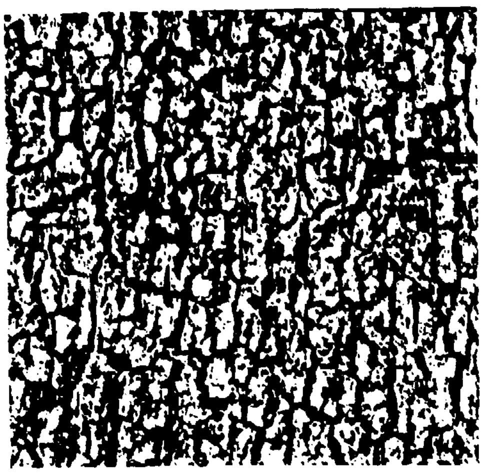


(a) Original

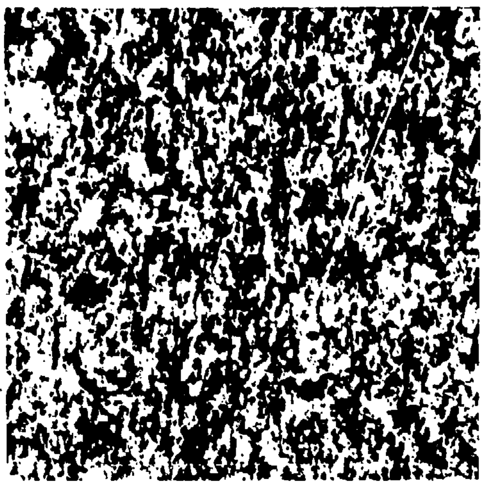
(b) N-gram

(c) Linear model

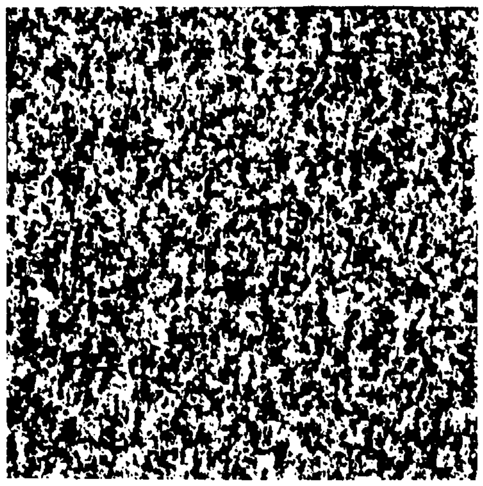
Fig. 16. Binary water.



(a) Original

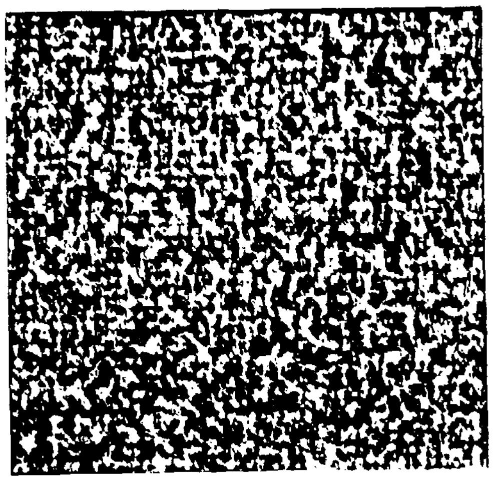


(b) N-gram

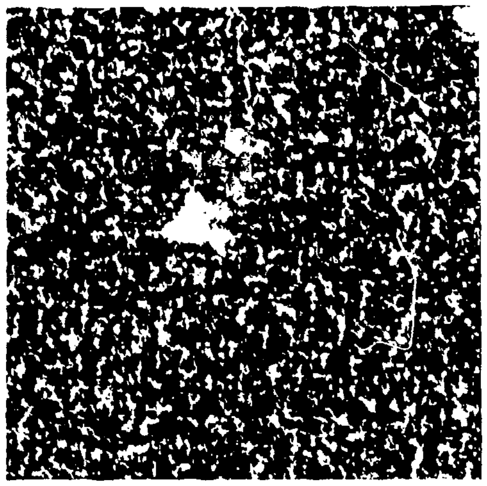


(c) Linear Model

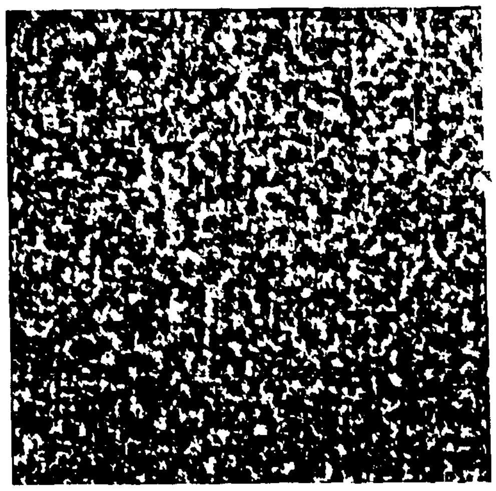
Fig. 17. Binary bark.



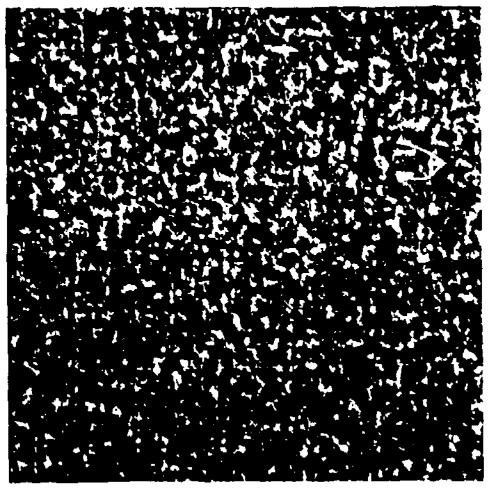
(a) Original



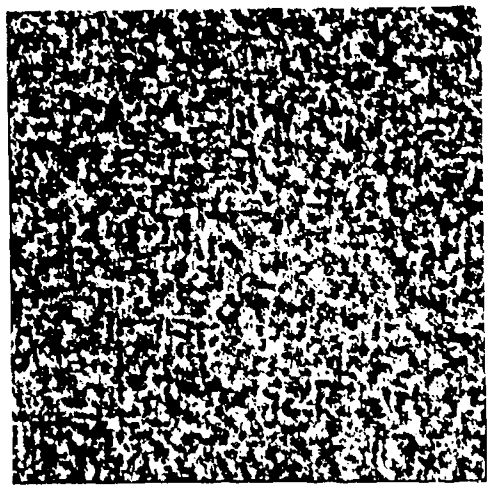
(b) First-order linear model



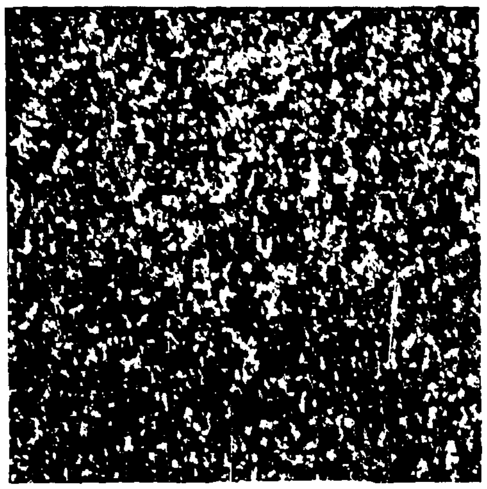
(c) Second-order linear model



(d) Second-order linear model with nonstationary noise

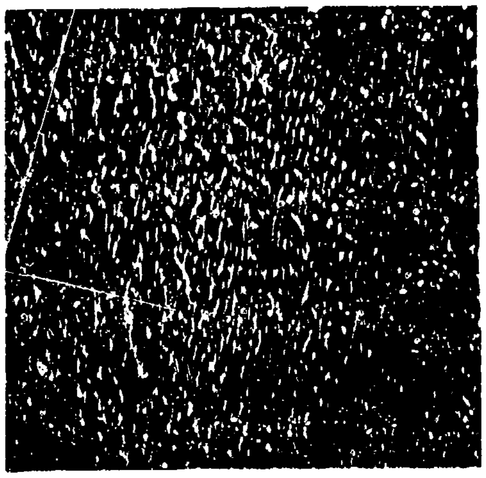


(e) Skip-generate model

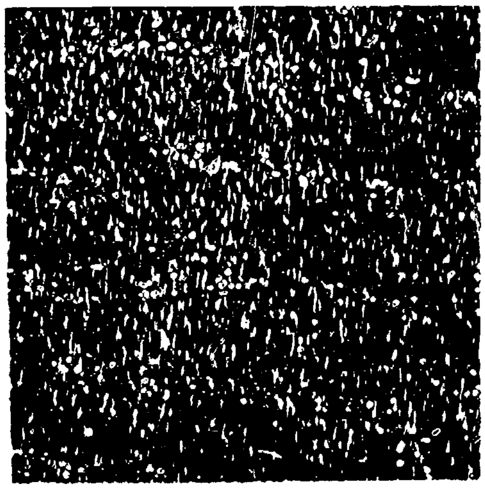


(f) Best-fit model

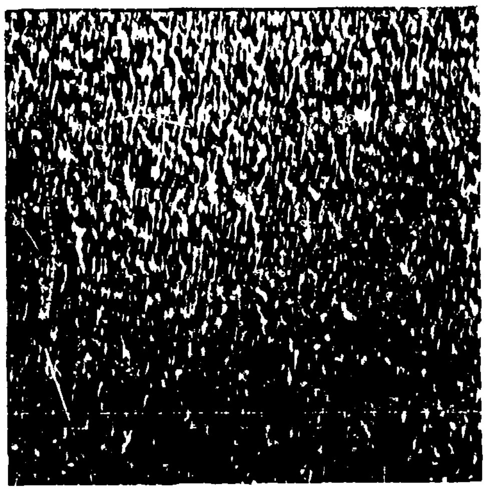
Fig. 18. Wool.



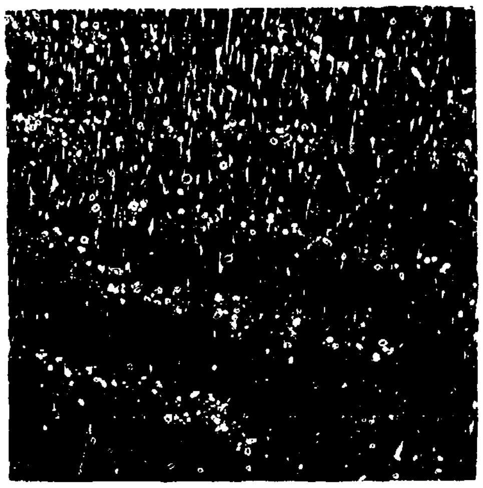
(a) Original



(b) First-order linear model

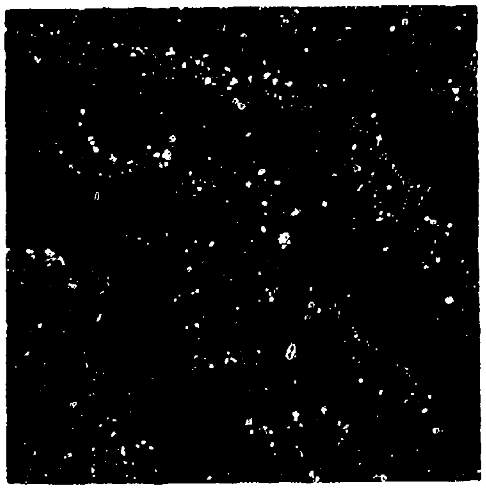


(c) Second-order linear model



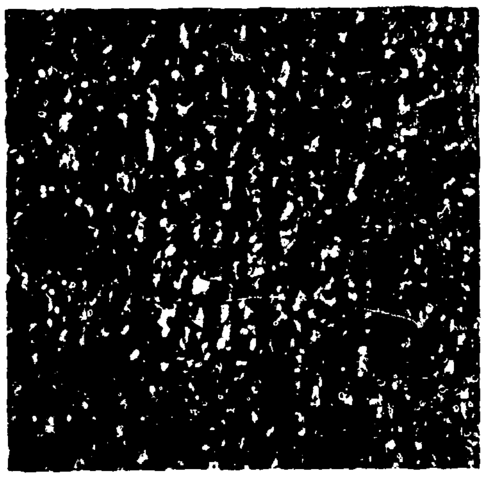
(d) Second-order linear model with nonstationary noise

(e) Skip-general model

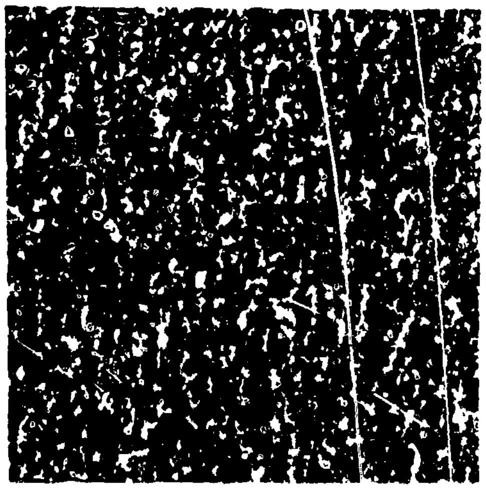


(f) Best-fit model

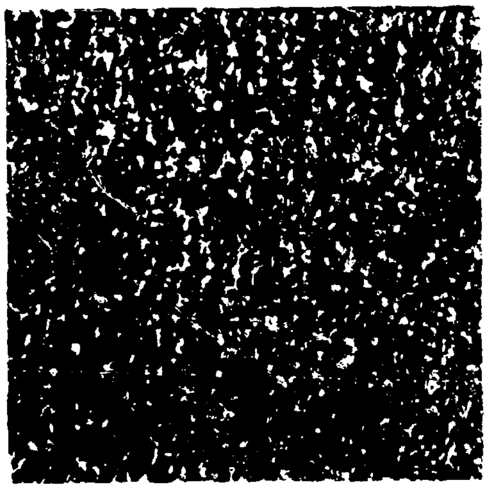
Fig. 19. Water.



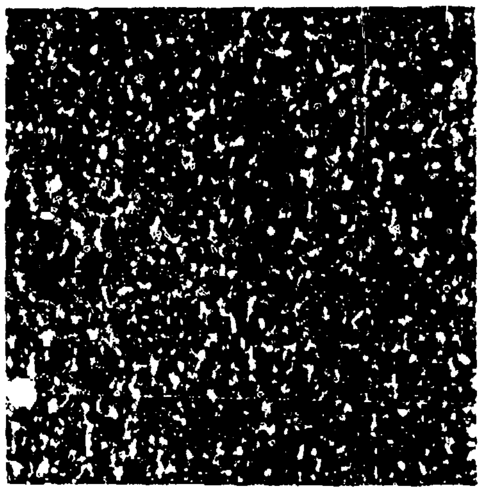
(a) Original



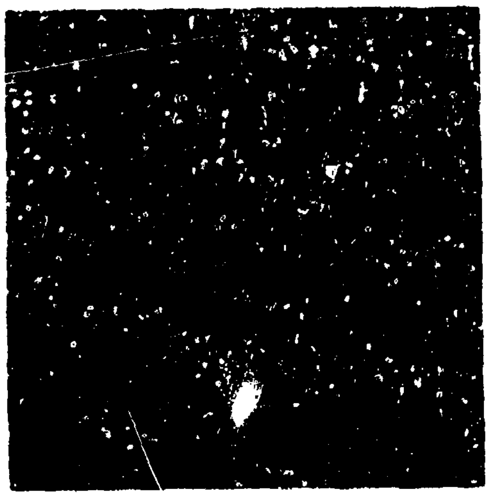
(b) First-order linear model



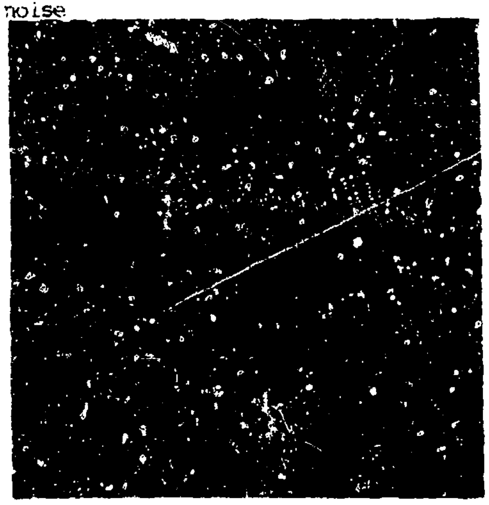
(c) Second-order linear model



(d) Second-order linear model with nonstationary noise



(e) Skip-generate model



(f) Best-fit model

Fig. 20. Bark.

TWO HIERARCHICAL LINEAR FEATURE REPRESENTATIONS: EDGE PYRAMIDS AND EDGE QUADTREES

Michael Shneier

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, MD 20742

ABSTRACT

Two related methods for hierarchical representation of curve information are presented. First, edge pyramids are defined and discussed. An edge pyramid is a sequence of successively lower resolution images, each image containing a summary of the edge or curve information in its predecessor. This summary includes the average magnitude and direction in a neighborhood of the preceding image, as well as an intercept in that neighborhood and a measure of the error in the direction estimate. An edge quadtree is a variable-resolution representation of the linear information in the image. It is constructed by recursively splitting the image into quadrants based on magnitude, direction and intercept information. Advantages of the edge quadtree representation are its ability to represent several linear features in a single tree, its registration with the original image, and its ability to perform many common operations efficiently.

INTRODUCTION

Edges provide much information about the contents of an image. Often this information is hard to interpret because of the large amounts of data involved and the existence of spurious edges that arise from noise in the image. This paper presents two related approaches to representing edges that attempt to overcome some of the difficulties in analyzing edges. The first approach is based on the use of a pyramid, or sequence of images, each a lower resolution version of its predecessor. The second involves a variable resolution representation, in which the local consistency of the edges determines the resolution at which they are represented. This approach builds a quadtree from the image, with leaves in the tree storing information about the edges that pass through square subregions of the image. Both of the representations are able to represent not only edges, but any linear information.

Many researchers have taken advantage of the pyramid structure to devise various efficient image-processing algorithms ([8], [14], [30], [31], [32]). Most of these algorithms, however, have dealt with images containing extended homogeneous regions, or blobs. The use of pyramids in linear feature analysis is significantly more complex than in region analysis. This is because a pyramid is well suited for representing images whose major features are two dimensional. Such features tend to retain their integrity and remain recognizable when lower resolution versions of the image are constructed using a simple rule such as averaging over small local neighborhoods. In contrast, the important features of edge or curve images are concentrated in a small proportion of the image, and it is the positions and orientations of the edges or curves that are the important information in the image. This paper provides a method of constructing edge pyramids that allows the advantages of the pyramid structure to be applied to edge and curve images. Some of these advantages include the compression of data to manageable size, and the ability to direct costly analysis in small regions of the original image, or set parameters such as thresholds. Projecting down from a given level in the pyramid also gives rise to an image in which all the features are of a known minimum size, and which has had much of the noise smoothed out.

A system has been implemented that builds pyramids from edge images, and can reconstruct edge images from low resolution levels in the pyramid. This system includes an edge enhancement scheme that is interesting in its own right. It also shows empirically the ability of an edge pyramid to retain most of the useful information at low resolutions, and the ability to reduce the amount of noise in the image.

The edge quadtree representation uses the same information as the edge pyramid, but has the ability to change the resolution of the representation to account for the edge information in the image. Thus, where edges are long and have consistent directions, large portions of the edges can be represented by leaves high up in the quadtree. Where edges are close together, however, or at corners, much smaller leaves may be needed to represent the edge information. This gives rise to a polygonal approximation of the curve information. The quadtree is shown to be useful for representing several edges or curves in a single structure, in contrast to other representations like the strip trees of Ballard ([1]), the upright rectangles of Burton ([4]), and the chain codes of Freeman ([18]). The representation allows many common edge operations to be performed efficiently, and the fact that it is in registration with the image, and with ordinary quadtree representations of region-like information built from the image, simplifies interaction between region and edge operations.

EDGE PYRAMIDS

Building an edge pyramid

A pyramid to be used in edge or curve analysis cannot simply be constructed by building an averaged pyramid and then applying an edge detector at each level. This is because smoothing in the pyramid might cause some edges to be missed, while those edges that are found will be displaced relative to the original image because the edge detector is operating on a different image. Were it not the case that direction information is crucial in edge analysis, a pyramid could be constructed from an edge magnitude image by using the maximum value of the magnitude in each neighborhood of the image as the value of a point in the successor level of the pyramid. Such an approach has been used to construct a pyramid of line information (Hanson & Riseman, [7]).

For the purposes of this discussion it is assumed that an edge image has been constructed with information stored at the pixels through which the edges pass. In order to construct an edge pyramid that preserves as much information as possible between levels, certain information must be present at each point of each level, as a summary of the information of the corresponding neighborhood of its predecessor. In the implementation to be described below, the neighborhoods were of size 4 by 4, with overlap between adjacent neighborhoods. The minimum information to be stored at each point is an estimate of the magnitude and direction of the edge(s) through the corresponding neighborhood. In addition, an intercept point is needed to fix the position of an edge in a neighborhood. A measure that is also useful is an indication of the error in the direction estimate. Errors will usually be high at corners or where more than one edge passes through a neighborhood. The error term can be used to signal such situations, and cause higher levels of the pyramid to ignore such regions. This gives rise to a class of edge quadtrees (Section 3).

Thus, it is necessary to provide a means of estimating the magnitude and direction of the edge(s) passing through a neighborhood, and an intercept point for each edge. To simplify matters, and to prevent the amount of information stored at each point from growing in an unbounded way, each neighborhood is restricted to having a single edge passing through it. Should more than one edge pass through a neighborhood, the best edge is sought using the following procedure.

The neighborhoods used in the implementation were of size 4 by 4, with each neighborhood sharing two rows with its vertical neighborbors (North and South of it), and two columns with its horizontal neighbors (West and East). The neighborhoods are shown in Figure 1. The method is based on the observation that the central 2 by 2 regions are disjoint and cover the picture. Thus, by first finding the best path through the central regions, and then extending it to the full 4 by 4 neighborhoods, the complexity of computation is significantly reduced.

Each point in an edge image contains two pieces of information, a magnitude and a direction. Within the central 2 by 2 region of each 4 by 4 neighborhood, the points having the maximum magnitude are found. Based on these values, a decision is made as to whether or not an edge exists in the neighborhood, and, if an edge exists, what kind of edge it is.

If the maximum value is greater than some minimum (currently 2 in the implementation) and the next to maximum value is greater than the minimum value, an edge with two points in the central region is assumed to exist unless the directions of the two points are not consistent (i.e., differ by more than 45 degrees). If one point is significantly greater than the rest in the central region, the assumption is made that an edge passes through the 4 by 4 neighborhood, but only touches one corner of the central 2 by 2 region. If no point differs significantly from its neighbors (i.e., by more than 2), no edge is assumed to pass through the 4 by 4 neighborhood. It would be represented, however, by the adjacent neighborhood through whose center it passed.

It must be understood that two kinds of direction information are used in evaluating edge continuation and edge consistency. First, there is the direction established by the edge detector as an estimate of the direction of the edge through a point. Second, there is the direction from a point in the grid of the image to another point. For example, a point has eight immediate grid neighbors, at angles of 0, 45, 90,..., degrees around it. These fixed angles, together with the directions calculated for edges at a point and its neighbors, are used to establish the continuity and consistency of neighboring edge points.

An edge with two points passing through the central 2 by 2 region may consistently be extended in three ways at each end (Figure 2a). The assumption made is that edges do not change direction too radically (i.e., by more than 45 degrees per pixel). Thus, instead of looking for all possible sequences of four points through a 4 by 4 neighborhood, two sets of three continuations are all that need be examined. The directions of these points are required to be compatible with the two-point edge for them to be considered as eligible for continuing the edge. Should more than one extension be found at each end, the best is chosen based on the grid angle between the points, their directions, and their magnitudes. The best extension at each end is used to adjust the magnitude and direction of the corresponding central edge point.

For one-point edges there are five possible extensions in the 4 by 4 neighborhood (Figure 2b). The best compatible extension is used to adjust the edge point here too. (Note that although only two extensions are compatible with the assumption that edges bend gradually, the other three points must be examined to allow for the case where an edge terminates inside the neighborhood). If no compatible edge extension can be found, the edge magnitude is decreased.

The above process is applied to all 4 by 4 neighborhoods in parallel. It may be iterated to allow information to propagate along the edges. The result is a preferred edge through each neighborhood. These chosen edges are used to construct the edge pyramid.

The process that actually constructs the pyramid is much simpler than that which establishes the best edges. For each neighborhood, it must calculate a magnitude, direction, intercept, and direction error value. The magnitude is calculated as the mean of the magnitudes in the 4 by 4 neighborhood. The direction is calculated as the mean of the directions of those points in the neighborhood with non-zero magnitude. The error term is the square root of the sum of squares of differences between the individual directions and the mean direction. The intercept is one of four values, denoting the position of the maximum magnitude point in the 2 by 2 central region of the neighborhood. The values are only calculated for a region if an edge actually passes through the central 2 by 2 region. These values are sufficient to reconstruct the edge to within the error tolerance. Other, more complex pyramid construction methods could be implemented. For example, it would be possible to use information high up in the pyramid to alter decisions made earlier. Because a decision about the edge through a quadrant is made based only on local information, it might be found that a different decision would have made the edges higher up in the pyramid more consistent. By backtracking to lower levels and altering the decisions made there, perhaps a more informed result would be obtained.

Note that more than one edge might pass through a neighborhood. In this case, it can be expected that the error term will be large, and the reconstruction less reliable. Reconstructing a level from its successor is also a simple process, although sophisticated and complex methods could be devised that would produce improved reconstructions. The method that was implemented makes no use of the error term, and does not require adjacent neighborhoods in the reconstructed image to be consistent. The process simply expands each point to a 4 by 4 neighborhood, and assigns the mean magnitude and direction stored at the point to each edge point in the expanded neighborhood. Points are chosen as edge points by requiring the edge to pass through the intercept point, and to lie along the assigned direction. The reconstructions that result are usually very reasonable, with the errors occurring only where there are sharp corners, or where edges are close together. The results can often be improved by applying the best edge routing discussed above.

Examples

The edge images for the examples presented below were obtained applying a zero-crossing edge detector (Marr & Hildreth, [11]) to gray level images. The edge detector returns magnitude and direction values for points corresponding to zero crossings in the Laplacian or second directional derivative of the image intensity. The zero crossings are approximated by the zero points of the difference between two Guassian-like functions with different standard deviations. The Laplacian is calculated using the hierarchical discrete correlation method of Burt ([3]). For each zero crossing point, a 5 by 5 Prewitt-like operator is used to give magnitude and direction information. The advantages of using a zero-crossing detector are that the edges are thin and the boundaries are closed curves. The edge pyramid process will, however, work for any edge or curve image.

The first example shows the entire process in a step by step way, using a binary image of a square. For more images, only the magnitude values produce meaningful pictorial displays, and only these images will be shown in later examples. In all the examples, the results of a 16-fold compression and a subsequent reconstruction are shown.

Figure 3a shows a binary image of a bright square of size 32 by 32 centered in a 64 by 64 image. Figure 3b shows the result of applying the zero-crossing edge detector to the image, while Figure 3c shows the image resulting from one iteration of the best edge procedure. In both cases, the top image is the direction image, and the bottom image is the magnitude image, both thresholded so that all non-zero points are displayed. After the best edge procedure has been applied, the directions of neighboring points are more consistent. This is the reason for the slight lengthening of the two short line segments at the bottom of the second direction image.

Figure 3d shows the four 32 by 32 images produced by the edge pyramid program. The images in the bottom row are the magnitude (left) and the intercept (right) of each point. Those in the top row are the direction image (left) and the error in direction (right). All images are thresholded so that non-zero points are displayed. The important thing to notice here is the error image. The only places at which errors in direction are detected are the corners.

Figure 3e shows the results of applying one iteration of the best edge procedure to the 32 by 32 images, and Figure 3f shows the 16 by 16 images produced by the pyramid process. The reconstruction algorithm is applied to the 16 by 16 images in Figure 3g, and to the reconstructed 32 by 32 images in Figure 3h. The results of the reconstruction illustrate the ability of the edge pyramid to retain full information in regions where there are long consistent lines. It is only at the corners that information is lost, and, in this case, a simple extension algorithm could be applied to restore the corners.

Figure 4 shows the results of running the whole process on the edges produced from a gray level image of a tank (Figure 4a). The original edge magnitude and the enhanced magnitudes are shown in Figure 4b. Figure 4c shows the first level of the pyramid, the best edges found at this level, and the second level of the pyramid. The first level of reconstruction and the final reconstructed image are shown in Figure 4d. Figure 4e shows the result of thresholding the magnitude images, all at the same threshold value. It is clear that the important information has been retained. A similar sequence is shown in Figure 5, starting from a gray level image of part of an airport.

Figure 6, finally, shows what happens when an image has sharp corners and inconsistent edges that occur close together. While the result of running the process is still clearly recognizable, the edges have been broken up into very short segments at the corners, and the reconstructed image is of clearly inferior quality to the original.

EDGE QUADTREES

A quadtree is obtained from a binary image by successive subdivision into quadrants. If the original image is homogeneous, a single leaf node is created. Otherwise, the image is divided into four quadrants, which become sons of the root node. This process is applied recursively until all terminal nodes are homogeneous. Binary quadtrees have been shown to be useful in representing large images compactly, and many algorithms have been developed for efficiently applying image processing techniques to images represented by quadtrees ([2], [5], [6], [9], [10], [19-29]).

For gray level images, a class of quadtrees can be defined, based on the brightness characteristics of the image. The root node represents the whole image, and typically stores the average gray level. If the image is sufficiently homogeneous (i.e., if the variance in gray level is not too great) no subdivision is performed. Otherwise, the image is divided into four subimages, and four children of the root are constructed. As long as the variance in any quadrant is higher than a threshold, the process is recursively applied. The result is a tree that represents the image to a degree of accuracy dependent on the threshold. By changing the threshold on the variance, a whole class of gray level quadtrees can be constructed. More generally, a class of quadtrees can be constructed using piecewise polynomial fits to the data in each quadrant. Gray level quadtrees are useful for image smoothing (Ranade & Shneier, [17]), shape approximation (Ranade _et al._, [15]), and segmentation (Ranade, [16] Wu _et al._, [33]).

Edge quadtrees are similar to the trees described above, except that the information stored at each node includes a magnitude, a direction, an intercept, and a directional error term. All the information about the edge that is stored is used in constructing the quadtree. As in the gray level quadtrees, a class of edge quadtrees can be defined based on the directional error term.

The construction of an edge quadtree proceeds by first examining the magnitude term, then the direction and direction error terms, and then the intercept term. If a node has a sufficiently low magnitude (i.e., no edge exists), then no subdivision is performed. Similarly, if the error term is sufficiently small, no subdivision is performed, with one exception, as follows. It may happen that a number of parallel edge segments run through a quadrant, so that the direction term is consistent with the data, and the error is low. Thus, a division based only on the error would not be performed. It is clear, however, that the quadrant does not represent the data at a sufficient level of detail to enable the set of parallel segments to be reconstructed. Thus, whenever the error term falls below threshold, a further check must be made to ensure that the intercept points in the quadrant are consistent (i.e., they lie along a line in the direction of the direction term). Should this not be the case, the quadrant must be subdivided. A final requirement for an edge quadtree is a flag that is turned on should an edge terminate within a quadrant. In this case, the intercept will be the point at which the edge terminates. The result of applying this process recursively to the image is a quadtree in which long edges that are nearly straight give rise to large leaves, or a succession of large leaves. Near corners, or where edges intersect, much smaller leaves are needed, perhaps as small as individual edge elements. A feature of edge images is the low percentage of points that contain interesting information. This means that an edge quadtree can be expected to contain many large leaves where there are no edges at all.

Figure 7 shows the edge magnitude image for the airplane picture of Figure 6, and the levels of the edge quadtree that are not empty (levels 0, 1, 2, and 3). Note that the leaves are upright square blocks in fixed positions and that the shape of the quadtree is dependent on the global co-ordinate system of the image. This is characteristic of all quadtrees. It is one of the major differences between this representation and the strip trees proposed by Ballard [1] (Section 4).

One of the advantages of the edge quadtree is that it can be used to represent an image that may contain more than one curve. Of course, a separate quadtree, perhaps smaller than the image quadtree, can be used to represent each curve, but it is preferable to use a single quadtree, both in order to maintain registration with the image and for compactness. Each curve in the tree can be named, and all terminal nodes representing part of a curve can be marked with the name of the curve. In addition, region-like information can be made available in the same structure, simplifying the interactions between regions and linear features. Notice that the quadtree for a closed curve and that for the region enclosed by the curve have closely related shapes.

If only one linear feature is represented by a quadtree, many operations become very efficient. If more than one curve is represented, however, some of the operations need to be done at a higher resolution, in order to ensure that the correct curves are involved. A way of alleviating this problem is to assign the names of curves passing through a quadrant to non-terminal as well as terminal nodes. A bound has to be put on the number of names allowed, however, because this may not be limited. All non-terminals that have more named curved segments passing through them than the bound allows can be flagged. When curve operations involving flagged nodes are executed, the descendants of the flagged nodes must be examined recursively to find the first one with the required names.

Many operations that are useful for manipulating edge and curve information can be implemented efficiently using edge quadtrees. Most of the algorithms are adaptions of quadtree algorithms for region representation. Only the broad outlines and necessary modifications will be given here.

First, an algorithm is presented for naming each curve in a quadtree. It is based directly on the algorithm for finding connected components of an image represented by a quadtree (Samet, [21]). First note that a leaf node can have no more than nine different curves passing through it. This is the number of "smooth" continuations through a sequence of three pixels, assuming less than a 90 degree change of angle between pixels. Thus the number of names at a leaf is bounded. (It is also necessary to assume that two or more curves cannot have arcs in common).

The algorithm involves three phases. The first pass assigns names to each curve node in the quadtree by starting at the North-West corner of the tree, and examining the South and East neighbors of the curve nodes. If the direction of a neighbor is compatible with the node (i.e., within the error tolerance) and its intercept is also compatible (i.e., lines up along the common direction with the node's intercept), the neighbor is given the same name. Otherwise, a new name is assigned. If a node is found that has already been named, and the node is compatible, an equivalence is established between the nodes.

The second phase processes the equivalent pairs to produce equivalence classes, while the third and final phase traverses the tree again, and assigns a single name to all members of an equivalence class. The names at the leaves of the tree can be propagated up to the non-terminal nodes at the same time, a nonterminal node being flagged if too many names are assigned to it.

Many operations commonly applied to linear data are facilitated by the quadtree representation. For example, to find the length of a curve segment, the tree is traversed starting at the root, and looking at nodes until the first leaf node belonging to the segment is found. The length contributed by this node is calculated as the length of a line through the intercept with direction given by the direction component, and bounded by the node's borders. To find the rest of the nodes in the curve, the FIND-NEIGHBOR and FIND-CORNER algorithms defined by (Samet, [27]) are used. They are applied on each side in the direction given by the node's direction component. Nodes that are further away from the leaf's direction than is allowed by the error tolerance can be ignored. The lengths of the nodes found in this way are added to the curve length, if they have the correct name. Each new node after the first will have at most one successor. When no more neighbors can be found, the length has been calculated. For closed curves, the original node must be flagged to ensure that the process terminates.

Other operations are easily defined as modifications of regular quadtree operations. The distance from a point to a curve can be calculated using a variant of the distance transform algorithm (Samet, [25]). Instead of finding the distance from a point (or the center of a BLACK node) to the nearest WHITE, or boundary point, the distance to the nearest curve point is found. Other algorithms, like union and intersection (Hunter & Steiglitz [9], Shneier [28]), require almost no alteration.

Interactions between region and boundary information are natural in this representation because of the registration of the images. Operations like the Superslice algorithm (Milgram [12]) can be performed on the quadtrees by taking advantage of the information contained in the shapes of the trees. The Superslice algorithm attempts to find the best segmentation of an image by matching edge and region information. A number of thresholds are applied to the image, giving rise to various new images. Each of these images is matched with the edge image, and that with the best region/boundary fit is chosen as the segmented image.

In the quadtree, this operation can be simplified and made more intelligent. Starting with the edge, and noting that the shape of the region quadtree is constrained by that of the edge quadtree, a class of candidate thresholds can be stored at each leaf node in the quadtree, each candidate giving rise to a region subtree with a shape consistent with the edge subtree. If, after all the nodes have had candidates assigned, there is a single threshold that gives the correct shape (i.e., the same threshold appears at all nodes), that threshold will give the required segmentation. Otherwise, a number of local thresholds may be applied to subimages, or an approximation to the segmentation can be made by choosing a compromise threshold.

COMPARISON WITH OTHER METHODS

Another hierarchical quadrant-based method for representing edges is that of Omolayole and Klinger ([13]). They recursively subdivide an edge image into quadrants down to a 2 by 2 level. A number of edge patterns are then sought in each subquadrant, and, if too few of these are found, the quadrant is discarded. The result is a kind of tree structure, with the leaves containing template-like representations of the edge data in them. The main aim of this method seems to be to discard the areas of the image that contain little or no information. For edge images, this can be expected to save fairly large amounts of storage. The edge quadtree differs from this approach in its treatment of quadrants containing edge information. Instead of having fixed-sized leaves, the quadtree allows leaves to be of the largest size consistent with the edge information they represent.

Other methods that have been devised for representing linear information are the upright rectangles of Burton ([4]), the strip trees of Ballard ([1]), and the chain codes of Freeman ([18]).

Freeman has developed one of the most compact and well-known boundary representations, called chain codes. These codes represent the relative grid positions of successive line points in a digital image. They are perhaps not as well suited to representing edge information as line information, but have the advantages of being compact and are not tied to any particular co-ordinate system.

Burton presented a method of representing polygonal lines using a series of upright rectangles. His work was extended by Ballard, who defined a representation for curve information called strip trees. A strip tree is a representation of a curve, obtained by successively approximating parts of the curve by enclosing rectangles. The structure is a binary tree, with the root node representing the bounding rectangle of the whole curve. This rectangle is broken into two parts at a point of maximum distance from the line joining the endpoints of the curve. The two parts are children of the root, and may be recursively subdivided until an error bound is satisfied. Note that the strip tree is not unique in cases where more than one extreme point exists.

All these other representations are able to represent only single curves, while the edge quadtree representation is able to represent several curves in the same tree. The edge quadtrees and edge pyramids are also in registration with the image, and with region-based representations like ordinary quadtrees and pyramids. This gives them a further advantage over the other representations. Where the other methods, and particularly the chain code, gain over the edge quadtree is in compactness, although the edge quadtree is actually storing more information than the other methods, and may give rise to better reconstruction of edge information.

DISCUSSION

There are two main reasons for developing pyramids for linear features. The first is to provide compression of the data, for example to allow linear features to be detected from low resolution images in a hierarchical image data base, thus reducing the number of full resolution images that need to be examined. The second reason is to enable the most prominent edge features (or the edge features larger than a given size) to be extracted from the image, and to discard smaller features.

It would be impractical to search a large image data base for the existence of an object with a set of known features. Rather, it would be useful to be able to filter out most of the images on the basis of gross tests, leaving only a few to be examined more closely. For region- or blob-like features, this ability can be provided by gray-level pyramids or quadtrees. While the most natural form for storing linear information is probably a linked list, it is desirable for uniformity to store linear feature information in a similar way as regional feature information. This facility is provided by the edge pyramids and edge quadtrees presented in this paper. Of course the representations are useful not only for edges, but for any linear information.

The second reason for building an edge pyramid is to enable noisy edges and edges that are too small to be filtered out of the image. In fact, this is achieved in two ways, both through the lateral best edge process and through the pyramid process. The best edge process is not designed specifically to enhance good edges and suppress bad ones, but it has this effect except where two edges intersect or two edges pass very close to each other. The pyramid process causes short segments to be lost high in the pyramid because, after a while, they fail to find good continuations. Note, though, that short broken edge segments could be joined together if the gaps were sufficiently small relative to the image resolution. The main requirments for an edge to continue to exist at successively higher levels in the pyramid are consistency and good continuation.

CONCLUSIONS

Two related representations for linear information have been presented. Edge pyramids have been shown to be able to store the important edge information in an image, even at fairly low resolution, with the ability to reconstruct images that look very much like the originals. The main loss in information is at intersections of edges, or where edges pass close to each other. Small edges, usually representing noise, are also lost.

Edge quadtrees have been presented as an alternative hierarchical representation for linear feature information, with the ability to represent the information at variable resolution, depending on the local consistency of the data. The advantages of edge quadtrees over other representations are their ability to represent more than one curve in a single structure, and their registration with the image and with other region-based representations for images.

REFERENCES

1. D. H. Ballard, Strip trees, a hierarchical representation for curves. Proc. DARPA Image Understanding Workshop, SRI International, April 1979, 121-133.

2. P. J. Burt, Tree and pyramid structures for coding hexagonally sampled binary images. TR-814, Computer Science Center, University of Maryland, College Park, MD, October 1979.

3. P. J. Burt, Fast, hierarchical correlations with Gaussian-like kernels. TR-860, Computer Science Center, University of Maryland, College Park, MD, January 1980.

4. W. Burton, Representation of many-sided polygons and polygonal lines for rapid processing. CACM 20, 3, March 1977, 166-171.

5. C. R. Dyer, A. Rosenfeld, and H. Samet, Region representation: boundary codes from quadtrees. TR-732, Computer Science Center, University of Maryland, College Park, MD, February 1979.

6. C. R. Dyer, Computing the Euler number of an image from its quadtree. TR-769, Computer Science Center, University of Maryland, College Park, MD, May 1979.

7. A. R. Hanson and E. M. Riseman, Processing cones: a parallel computational structure for scene analysis. COINS working paper, University of Massachusetts at Amherst, 1976.

8. A. R. Hanson and E. M. Riseman, Segmentation of natural scenes. In Computer Vision Systems, (Hanson and Riseman, eds), Academic Press, New York, 1978.

9. G. M. Hunter and K. Steiglitz, Operations on images using quad trees. IEEE Trans. PAMI-1, 2, April 1979, 145-153.

10. A. Klinger and C. R. Dyer, Experiments in picture representation using regular decomposition, Computer Graphics Image Processing 5, 1976, 68-105.

11. D. Marr and E. Hildreth, Theory of edge detection. AI Memo 518, MIT AI Laboratory, April 1979.

12. D. L. Milgram, Region extraction using convergent evidence. Computer Graphics Image Processing 11, 1979, 1-12.

13. J. O. Omolayole and A. Klinger, A hierarchical data structure scheme for storing pictures. In Pictorial Information Systems (S. K. Chang and K. S. Fu, eds), Springer 1980.

14. T. Pavlidis, Structural Pattern Recognition, Springer 1977.

15. S. Ranade, A. Rosenfeld, and H. Samet, Shape approximation using quadtrees. TR-847, Computer Science Center, University of Maryland, College Park, MD, December 1979.

16. S. Ranade, A. Rosenfeld, and J. M. S. Prewitt, Use of quadtrees for image segmentation. TR-878, Computer Science Center, University of Maryland, College Park, MD, February 1980.

17. S. Ranade and M. Shneier, Using quadtrees to smooth images. TR-894, Computer Science Center, University of Maryland, College Park, MD, April 1980.

18. A. Rosenfeld and A. C. Kak, Digital Picture Processing, Academic Press, New York, 1976.

19. H. Samet, Region representation: Quadtrees from boundary codes. TR-741, Computer Science Center, University of Maryland, College Park, MD, March 1979.

20. H. Samet, Computing perimeters of images represented by quadtrees. TR-755, Computer Science Center, University of Maryland, College Park, MD, April 1979.

21. H. Samet, Connected component labeling using quadtrees. TR-756, Computer Science Center, University of Maryland, College Park, MD, April 1979.

22. H. Samet, Region representation: raster-to-quadtree conversion. TR-766, Computer Science Center, University of Maryland, College Park, MD, May 1979.

23. H. Samet, Region representation: quadtrees from binary arrays. TR-767, Computer Science Center, University of Maryland, College Park, MD, May 1979.

24. H. Samet, Region representation: quadtree-to-raster conversion, TR-768, Computer Science Center, University of Maryland, College Park, MD, June 1979.

25. H. Samet, A distance transform for images represented by quadtrees. TR-780, Computer Science Center, University of Maryland, College Park, MD, July 1979.

26. H. Samet, A quadtree medial axis transform. TR-803, Computer Science Center, University of Maryland, College Park, MD, August 1979.

27. H. Samet, Neighbor finding techniques for images represented by quadtrees. TR-857, Computer Science Center, University of Maryland, College Park, MD, January 1980.

28. M. Shneier, Linear time calculations of geometric properties using quadtrees. TR-770, Computer Science Center, University of Maryland, College Park, May 1979.

29. M. Shneier, Path length distance transforms for quadtrees. TR-794, Computer Science Center, University of Maryland, College Park, MD, July 1979.

30. M. Shneier, Using pyramids to define local thresholds for blob detection. Proc. DARPA Image Understanding Workshop, University of Southern California, November 1979, 31-35.

31. S. L. Tanimoto, Regular hierarchical image and processing structures in machine vision. In Computer Vision Systems (Hanson and Riseman, eds), Academic Press, New York, 1978.

32. L. Uhr, "Recognition cones", and some test results; the imminent arrival of well-structured parallel-serial computers; positions, and positions on positions. In Computer Vision Systems (Hanson and Riseman, eds), Academic Press, New York, 1978.

33. A. Y. Wu, T-H. Hong, and A. Rosenfeld, Threshold selection using quadtrees, TR-886, Computer Science Center, University of Maryland, College Park, MD, March 1980.

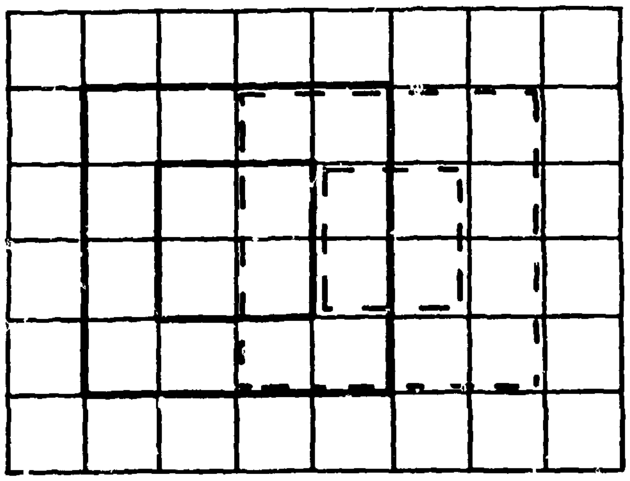
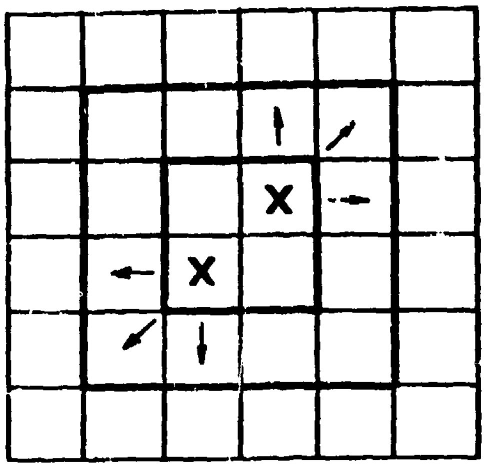


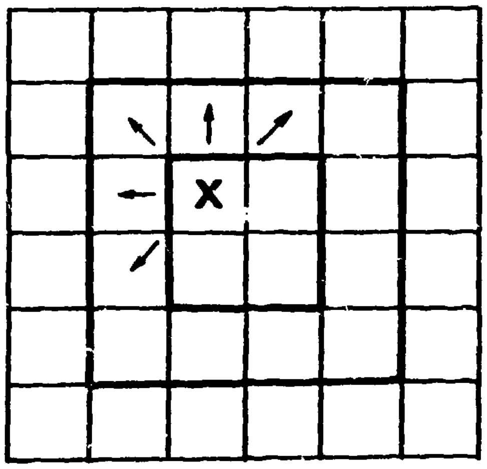
Figure 1. Two neighborhoods used in constructing pyramids. The central 2 by 2 regions are disjoint, but each neighborhood shares rows or columns with its neighbors.

ACKNOWLEDGEMENTS

The support of the U.S. Army Night Vision Laboratory and the Defense Advanced Research Projects Agency under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged. The author would like to thank Ann Scher for producing the diagrams.



(a)



(b)

Figure 2. The ways in which edge points may be extended. a. For a two-point edge, there are six consistent continuations. b. For a one-point edge, there are five consistent continuations.

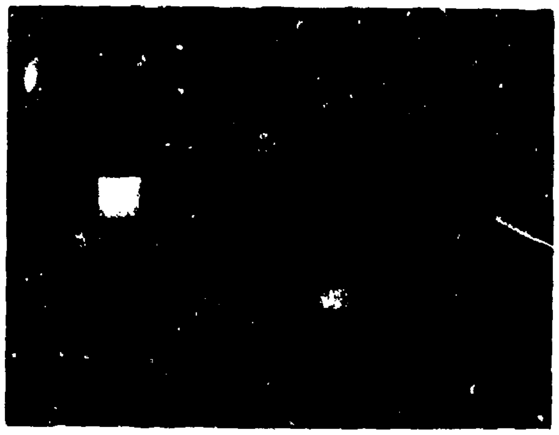


Figure 3:
(a) (b) (c)
(d) (e) (f)

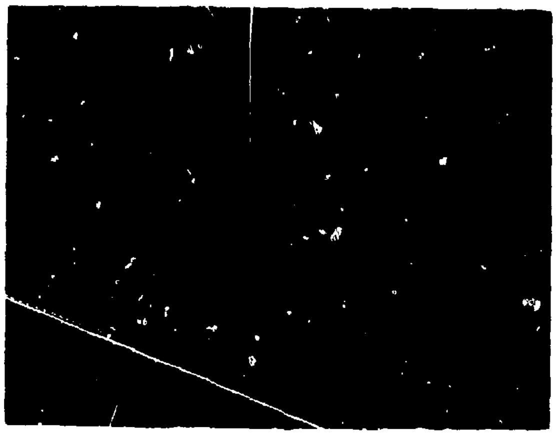


Figure 4:
(a) (b)
(c1) (c2) (d)

Figure 3:
(g) (h)

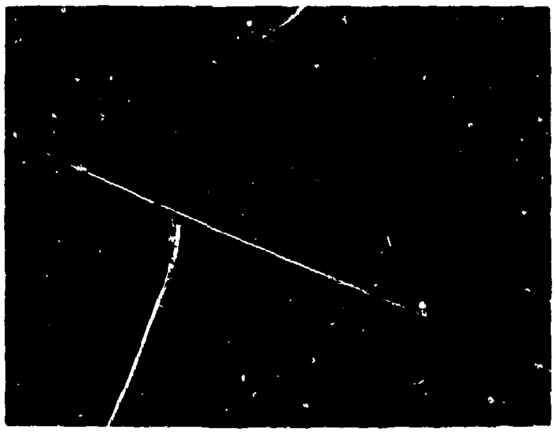


Figure 4e

Figure 3. The pyramid process applied to a 64 by 64 binary image. a. Original image of a square region. b. The magnitude (bottom) and direction (top) images produced by a zero-crossing edge detector. c. The magnitude and direction images after running the best edge procedure. d. The first pyramid level (32 by 32): magnitude (bottom left); intercept (bottom right); direction (top left); error (top right). e. The result of applying the best edge procedure to the 32 by 32 images. f. The second pyramid level (16 by 16). g. The result of reconstructing a 32 by 32 image from the 16 by 16 pyramid level. h. The result of constructing a 64 by 64 image from the 32 by 32 reconstructed image.

Figure 4. The pyramid process applied to a gray level image. a. A FLIR image of a tank. b. The magnitude and enhanced magnitude of the edge image of the tank (thresholded so that non-zero points are displayed). c. The first pyramid level magnitude, the enhanced magnitude, and the second pyramid level magnitude. d. The first and second level reconstructed images. e. The same process as in a and b above, but with all images thresholded at the same level.

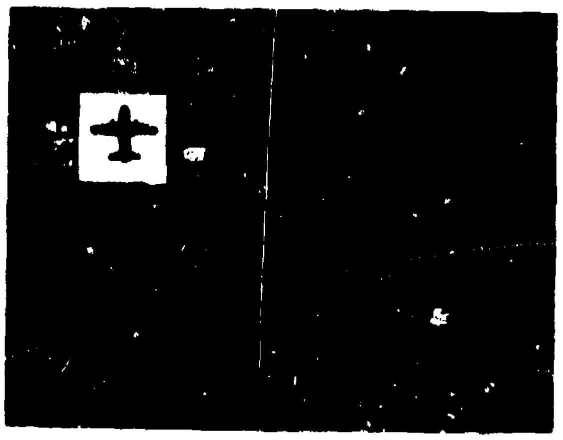


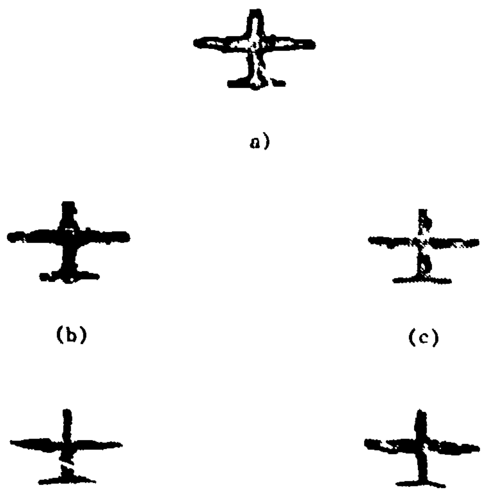
Figure 5:

(a) (b)

(c1) (c2) (d)

Figure 6. The pyramid and reconstruction process applied to a binary image of an airplane.

Figure 5e



a)

(b) (c)

(d) (e)

Figure 5. The process described in Figure 4 applied to an image of part of an airfield.

Figure 7. The edge quadtree of an airplane image. a. The edge magnitude image. b. The lowest level (level 0) of the edge quadtree (individual pixels). c. Level 1, having 2 by 2 blocks of pixels. d. Level 2, having 4 by 4 blocks. e. Level 3, having 8 by 8 blocks.

EDGE EVALUATION USING LOCAL EDGE COHERENCE

Les Kitchen
Azriel Rosenfeld

Computer Vision Laboratory, Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

A method of evaluating edge detector output is proposed, based on the local good form of the detected edges. It combines two desirable qualities of well-formed edges -- good continuation and thinness. The measure has the expected behavior for known input edges as a function of their blur and noise. It yields results generally similar to those obtained with measures based on discrepancy of the detected edges from their known ideal positions, but it has the advantage of not requiring ideal positions to be known. It can be used as an aid to threshold selection in edge detection (pick the threshold that maximizes the measure), as a basis for comparing the performances of different detectors, and as [illegible] of the effectiveness of various types of preprocessing operations facilitating edge detection.

INTRODUCTION

The concept of an <u>edge</u> is a difficult one to define precisely. The stimulus conditions that cause the perception of an edge by humans are by no means simple to describe. There are many well known visual paradoxes in which an edge is clearly seen where none physically exists. (See for example Cornsweet [1974], Dember [1966], or Gregory [1974].) In the analysis of images by computer, exactly what constitutes an edge depends greatly on the objectives of the analysis.

Keeping the above in mind, we can nonetheless regard an <u>edge</u> as the boundary between two adjacent regions in an image, each region homogeneous within itself, but differing from the other with respect to some given local property. Thus an edge should ideally be line-like.

In this paper we restrict our attention to the simplest case, brightness edges, although the edge evaluation techniques we present below are applicable to color or texture edges as well. Brightness edges in an image have many possible causes in the original scene: discontinuities in surface properties (such as reflectance), in surface orientation, in illumination (shadows, for example) or in depth (causing occlusion of one surface by another). However, the interpretation of the cause of an edge will not concern us here.

Brightness edges (henceforth just <u>edges</u>) are important features of image analysis, and, accordingly, many schemes have been devised for detecting them. Here we are concerned chiefly with so-called enhancement/thresholding edge detectors: In the enhancement step, an operator which computes local brightness differences is applied to an image. Such an operator will have a high response when positioned on the boundary between two regions, but little or no response within each region. (The operators discussed below also compute an estimate of the direction of brightness change.) In the next step, the edges in the image are extracted by suitably thresholding the operator output. The final result of processing is a binary picture, pixels deemed to be on an edge (<u>edge pixels</u>) having the value 1, all others (<u>non-edge pixels</u>) have the value 0.

It is of interest to evaluate the quality of the output of an edge detector, both to compare one detector scheme with another, and also to study the behavior of a given detector under different conditions and parameter settings. Several authors have proposed techniques for edge evaluation. In the next section we review their work.

SURVEY OF PREVIOUS WORK

Fram and Deutsch (1974, 1975) studied the effect of noise on various edge detector schemes. For this purpose they used synthetic images composed of three vertical panels. The outer panels were of two different grey levels; the narrow inner panel interpolated between these grey levels. It was considered that the position of the edge was defined by this central panel and only here should an edge detector respond. Images were generated for a number of different levels of contrast between the two outer panels, and to each image was added identically distributed zero-mean Gaussian noise.

Several different edge enhancement techniques were applied to these test images, and thresholds were chosen so that the number of detected edge points was as close as possible to the number of points expected for a well-found edge, based on inspection of a sample of detector outputs. The threshold output was evaluated according to two measures. The first, P_1, estimated what fraction of the detected edge pixels were actually edge points. The second, P_2, estimated what fraction of the vertical extent of the edge was covered by detected edge pixels. These estimates are possible because it was known that edge pixels actually due to the edge could be found only in the central panel, and it was assumed that edge pixels due to noise could be uniformly distributed throughout the image.

As would be expected, the experimental results showed that edge detector performance, as measured by P_1 and P_2, improves when contrast is increased relative to noise. They also demonstrated that some edge detection schemes perform consistently better than others.

While their measures are directly applicable only to vertical edges, Fram and Deutsch also performed experiments with synthetic oblique edges. They did this by the expedient of numerically rotating the enhancement output until it corresponded to a vertical edge. It could then be thresholded and evaluated as if it had been vertical. By this means they examined the sensitivity of the detectors they used to edge orientation.

The approach of Abdou and Pratt [1979] is more analytic. (See also Abdou [1978].) Using a simple model for the digitization of a straight edge passing through the center of an operator's domain, they goemetrically analyzed the sensitivity of a number of edge enhancement operators to the orientation of the edge. They similarily analyzed the fall-off of operator response with displacement from the center of the domain for straight edges with vertical and diagonal orientations.

They described a statistical design procedure for threshold selection in noisy images with vertical and diagonal edges. Using additive Gaussian noise as an example, they derived the conditional probability distributions of operator response for a number of enhancement operators, given the existence or non-existence of an actual edge. They could thus compute for each operator the probabilities of correct and false detection as a function of threshold and of noise level. By this means they showed the superiority of some detection schemes over others. They also presented a pattern-classification approach to threshold selection using training samples of edge and no-edge neighborhoods, and gave experimental results for a number of edge detectors in discriminating edge from non-edge neighborhoods, using this approach. These results show a similar ordering of the quality of the various edge detection schemes.

More relevant to the present paper, Abdou and Pratt provided another experimental comparison of the various edge detector schemes using Pratt's figure of merit of edge quality [Pratt 1978]. They used synthetic test images very similar to those of Fram and Deutsch above. The only difference worth remarking on is that Abdou and Pratt vary the relative strength of signal to noise by holding the contrast constant and changing the standard deviation of the added noise. Pratt's figure of merit is based on the displacement of each detected edge pixel from its ideal position (known from the geometry of the synthetic image), with a normalization factor to penalize for two few or too many edge points being detected. Its definition is:

$$F = \frac{\displaystyle\sum_{i=1}^{I_A} \frac{1}{1 + \alpha(d(i))^2}}{\max\{I_A, I_I\}}$$

where I_A is the actual number of edge pixels detected; I_I is the ideal number of edge pixels expected (known from the geometry of the synthetic image); d(i) is the miss distance of the ith edge pixel detected; and α is a scaling factor to provide a relative weighting between smeared edges, and thin, but offset edges. For these experiments, Abdou and Pratt set $\alpha = 1/9$. Like Fram and Deutsch's parameters P_1 and P_2, this figure of merit was implemented for vertical edges, but Abdou and Pratt also present a modification of it for diagonal edges.

Unlike Fram and Deutsch, Abdou and Pratt used the less arbitrary procedure of choosing thresholds so as to maximize the figure of merit. The experimental results showed, as one would expect, that the figure of merit declines with increased noise, and also again showed the superiority of certain edge detection schemes over others.

The work of Bryant and Bouldin [1979] is different in several respects. They used real aerial photographs instead of synthetic images. Their threshold selection was based on accepting a fixed upper percentile of the distribution of enhanced edge output. More significantly, they proposed two quite distinct edge evaluation measures. One, called absolute grading, is based on the correlation of the edge detector output with an ideal "key" output, this key being determined apparently by hand. Their other technique, called relative grading, is rather novel. Omitting the details, it is based on comparing the output of a number of detectors, and rating each detector by how often it agrees with the consensus of the other detectors in deciding whether an edge exists at each pixel. By these means they compared a number of edge detectors, and were able to some extent to quantify the improvement in edge output achieved by such post-processing as edge-linking and edge-thinning. They also gave an example of effect of threshold level on the absolute grade of an edge detector.

Relative grading, while an interesting idea, suffers from a number of weaknesses. Its results depend on the details of the consensus determination used, and on the particular mix of operators chosen for comparison. Most important, it is completely oblivious to detection errors made by all detectors, and may even penalize a good detector that does not make an error made by a majority of bad detectors.

Aside from relative grading, all methods discussed above require prior knowledge of the location of the actual edge, since they are more or less based upon the discrepancy between the detected edge pixels and the ideal position of the edge. This is fine for experiments with controlled synthetic images, but raises questions when applied to real images, since the determination of edges in such pictures is very much the subjective decision of a human observer. The techniques are completely inapplicable to images for which the actual edge locations are unknown.

Further, the discrepancy between detected and ideal edge is not the sole determinant of the quality of edge output. See Figure 1. Here we have two detected edges, both of equal discrepancy from the ideal. However, one of them is clearly preferable, since the detected edge is continuous, rather than fragmented. It is clear that some attention should be paid to the good form of the detected edge.

Finally, none of the above edge evaluation measures take any account of the edge direction information produced by most edge enhancement operators. This information is used in many applications and is an important consideration in determining the good form of an edge. Even though a set of edge pixels may lie in the shape of a well-formed edge, something is amiss if the estimated edge directions are chaotic. Ideally, the brightness gradient direction should be everywhere perpendicular to the edge, and perpendicular in the same sense.

LOCAL EDGE COHERENCE

Bearing in mind the deficiencies of the above techniques, we have developed an edge evaluation measure based solely on the criterion of good edge formation, without using any prior knowledge of ideal edge location. This new measure is intended as a supplement to existing measures, not a replacement, since it is clear that a measure which disregards the correct location of an edge cannot be a fully adequate measure (although the results presented below show that it is quite good). For example, an edge detector that systematically mislocates edges will, by our schemes, receive an evaluation measure equal to that of a detector which perfectly locates edges.

However, since the new measure does not require prior knowledge of edge location, it can be used much more freely, in particular on images for which this knowledge is lacking. In addition to the standard uses of comparing edge detector schemes, the new measure can be used for selecting and adjusting edge operators as they are applied to an actual image. For example, an edge detector threshold can be chosen so as to maximize the edge evaluation measure. This will be the threshold which extracts the best-formed edges. (This parallels the work of Weszka and Rosenfeld [1978] on threshold evaluation for segmentation of regions. One of their techniques rated a threshold level on the basis of the busyness of the resulting thresholded image.) In applications where edge extraction is an important part of the processing, the edge evaluation measure can serve as an indication of image quality.

The approach we have used is based on what we call local edge coherence. Essentially, we examine every three by three neighborhood of the thresholded edge output, taking into account the direction output as well. If the center of the neighborhood is an edge pixel, then we call the neighborhood an edge neighborhood and rate it on the basis of two criteria, continuation and thinness, which should both be exhibited by a well-formed edge passing through the center of the neighborhood. Both these criteria are based on the working definition of an edge given in Section 1. It should be locally line-like, with due regard for the consistency of direction of brightness changes. Continuation requires, ideally, that adjacent to the central pixel, along the edge (this is perpendicular to the gradient direction of brightness change at the center), there be two edge pixels with almost identical direction which form the continuation of that edge. Thinness requires, ideally, that all the other six pixels of the neighborhood be no-edge pixels. The continuation and thinness ratings of an entire edge output can be measured as the fraction of edge neighborhoods satisfying these respective criteria.

Of course, for most images, very few edge neighborhoods will perfectly satisfy these two criteria, because of digitization problems and even slight noise. We therefore compute instead continuation and thinness scores, ranging from 0 to 1, with the overall scores being averaged over every edge neighborhood in the output. These scores are designed to take the value 1 for perfectly-formed edge neighborhoods, dropping off only slightly for almost well-formed neighborhoods, but falling eventually to 0 for badly formed neighborhoods.

The continuation score is computed as follows:

Let $|\alpha-\beta|$ represent the absolute difference between two angles α and β, the difference ranging from 0 to π radians. Let

$$a(\alpha,\beta) = \frac{\pi-|\alpha-\beta|}{\pi}$$

This function ranges from 1 for identical angles α and β, linearly down to 0 for angles that differ by half a revolution, that is, point in opposite directions. It thus measures the extent to which the two angles agree in direction.

Let us number the neighbors of an edge pixel as shown in Figure 2. Let d stand for the edge gradient direction at the center pixel, and let $d_0, d_1, \ldots, d_7$ stand for the edge gradient directions at each of the eight neighbors respectively. Let

$$L(k) = a(d,d_k)\,a(\tfrac{\pi k}{4}, d+\tfrac{\pi}{2}) \text{ if neighbor k is an edge pixel}$$
$$= 0 \qquad \text{otherwise.}$$

This function measures how well a neighboring pixel continues on the left of the edge which passes through the central pixel. It is 0 when the neighbor is not an edge pixel, since no continuation exists. When the neighbor is an edge pixel, its rating is composed of two factors: The first, $a(d,d_k)$, measures how well the edge gradient direction at the neighbor agrees with that at the center. The second factor,

$$a(\tfrac{\pi k}{4}, d + \tfrac{\pi}{2})$$

measures how close neighbor k is to the expected direction of leftward continuation of the edge, based on the direction at the center. The term $\frac{\pi k}{4}$ is the direction to neighbor k, and the term $d+\frac{\pi}{2}$ is at right angles to the gradient direction and therefore lies along the edge. Similarly we define

$$R(k) = a(d,d_k)\cdot a(\tfrac{\pi k}{4}, d-\tfrac{\pi}{2}) \text{ if neighbor k is an edge pixel}$$
$$= 0 \qquad \text{otherwise}$$

which measures how well neighbor k continues the edge toward the right.

Of the three neighboring pixels lying to the left of the central edge gradient direction, the one with the highest value of L(k) is taken as the left continuation. Similarly, of the three pixels on the right, the one with the best value for R(k) is taken as the right continuation of the edge. The average of these two best continuations is taken as the continuation measure C for the entire neighborhood.

The thinness measure T for the neighborhood is computed as that fraction of the remaining six pixels of the neighborhood which are non-edge pixels. This will range from 1 for a perfectly thin edge, down to 0 for a very blurred edge.

Neither of these measures is independently useful for edge evaluation, as will be explained below. However a linear combination of the two

$$E = \gamma C + (1-\gamma)T$$

serves quite well for suitable values of γ. This parameter γ can be adjusted to give a relative biasing of the measure E in favor of well-connected edges as against thin edges. The choice of γ will also be discussed below.

While this approach to edge evaluation is a little *ad hoc*, no simpler technique seemed able to capture the notion of a locally well-formed edge. We were first led to investigate the possibility of an edge evaluation measure based on good form by an observation on compatibility coefficients for relaxation labelling [Peleg and Rosenfeld, 1977]. The arrays of compatibility coefficients showed a particular diagonal tendency when derived from images with clear edges which was far less pronounced when derived from noisy or blurred images. We attempted to devise an edge evaluation measure based on characteristics of the compatiability coefficient arrays, and later on characteristics of the edge direction co-occurrence matrices, which are closely related. Preliminary experiments showed that none of these measures were satisfactory, although they suggested that a measure based on good form could ultimately be developed. Several techniques based on local properties of the edge output were investigated, culminating in the method presented here. This measure is intuitively reasonable, and more important, performs quite well, as the experimental results below demonstrate.

One defect of our measure (though shared by all others) is that it can only be applied after thresholding. We endeavored to remedy this by devising methods that treat all pixels as potential edge pixels, but weigh their contributions by a function of their edge magnitudes. Unfortunately, the enormous number of low-magnitude pixels distorts the measure, unless the weighting function is of such a form as to be tantamount to thresholding.

It should be pointed out that this approach can be easily adapted to measuring the good form of other features, such as lines or corners, which are normally detected by some sort of template matching.

EXPERIMENTS

We present here some experiments which investigate the behavior of a number of edge detection schemes under various condtions. To permit a comparison, we have tried to make our experimental setup as similar as possible to that of Abdou and Pratt. We have used the same edge detection schemes (although our measure also makes use of edge direction information), the same noise model (additive, independent zero-mean Gaussian noise), the same threshold selection criterion (choosing that threshold which maximizes the evaluation measure), and for one series of experiments, essentially the same test image.

Test Images and Edge Detectors Tested

Two test images of edges were used: the first, 64 by 64 pixels, consisted of a left panel with grey level 115, a right panel with grey level 140, and a single central column of intermediate grey level

128. This we will call the "vertical edge" image. It is virtually the same as one of the test images used by Abdou and Pratt. In order to present conveniently edges at all orientations, we chose a second test image consisting of concentric light rings (grey level 140) on a dark background (grey level 115). This image was originally generated as a 512 by 512 image, with a central dark circle of radius 64, surrounded by three bright rings of width 32, these being separated by two dark rings of the same width, with a dark surround. The decision as to whether a pixel should be light or dark was based on its Euclidean distance from the center of the image. Then this image was reduced to size 128 by 128, by replacing each 4 by 4 block with a single pixel having the average grey level of the block. The reduction gave a convenient way of approximating, for curved edges, the digitization model used by Abdou and Pratt. We call this the "rings" image. While the edges in this test image are curved, they are locally almost straight, at all possible orientations.

To study the effects of noise, independent zero-mean Gaussian noise was added to each of the test images at seven different signal to noise ratios: 1,2,5,10,20,50 and 100. Following Pratt, the signal to noise ratio (SNR) is defined to be

$$SNR = \frac{h^2}{\sigma}$$

where h is the edge contrast (in this case 25), and σ is the standard deviation of the noise, adjusted to give the selected values of SNR. As an extreme case, we used an additional 64 by 64 test image with no well-formed edges, just Gaussian noise with mean 128 and standard deviation 16.

Figure 3 shows the vertical edge image, noise free as well as with the various levels of added noise. Figure 4 shows the same for the rings image. At the higher signal to noise ratios, the noise is almost imperceptible to the human eye. However, it is quite significant to the edge detectors used, since all of them have only small domains.

Ten different edge enhancement schemes were tested. The first group are the so-called "differential" operators. These measure the horizontal and vertical components of the brightness change by applying a pair of linear masks. The edge gradient direction is computed from these two components using the inverse tangent; and the edge gradient magnitude is computed either as the square root of the sum of squares of the two components, or as a sum (or max) of absolute values, for computational simplicity. Three different pairs of masks were used: those defined by Prewitt, Sobel and Roberts. Since the edge magnitude can be computed in two ways, this gives six methods altogether. The second group are the "template-matching" operators: three-level, five-level, Kirsch and compass-gradient. Each of these applies eight masks at every neighborhood. The edge magnitude is taken to be the strongest response out of these eight masks, and the edge gradient direction is given by the preferred orientation of the strongest-responding mask. For details on and references to all these operators, see Abdou and Pratt [1979].

Detailed Evaluation of One Detector

Before presenting an overall comparison of these edge detection schemes, we would like to examine in detail the results of the edge evaluation on a single scheme in order to discuss the properties of the edge evaluation measure itself. For this we have chosen the three-level template matching operator because it performed consistently better than any of the other operators in the comparison experiments described below. Even so, the results of the edge evaluation measure follow much the same pattern for the other operators as well.

Figure 5 shows the histogram of edge magnitude outputs for the three-level operator applied to the rings image with SNR 50, and Figure 6 shows the edge magnitude thresholded at nine levels equally spaced through its range. In Figure 7 are shown plots of edge evaluation against threshold for various values of the weighting factor γ. Figure 8 shows the same data, but plotted instead against the fraction of pixels which are edge pixels at each threshold, scaled logarithmically. This is a better way of presenting the data, since it is the selection of edge pixels that really matters, not the threshold directly.

We see that the thinness measure alone (γ=0.0) is of little use for edge evaluation. It reaches its maximum value at high thresholds since it rates a set of isolated edge pixels higher than an even slightly blurred edge. On the other hand, the continuation measure performs reasonably well by itself (γ=1.0), reaching a maximum value at a threshold which selects quite a good set of edge pixels. (This peak is more pronounced in Figure 8, since changing the threshold near the maximum produces only a small change in the population of edge pixels. Notice that this threshold lies in the valley of the histogram.) However, a close examination shows that these edges are several pixels thick, Better results are achieved with a lower value of γ. For the rest of this paper we will use γ=0.8, since this value seems to give the best compromise between continuation and thinness.

Table 1 shows the maximum values of the edge evaluation measure, and the thresholds at which they occur, for the various values of γ. Figure 9 shows the thresholded edge magnitude for a range of closely spaced thresholds around that at which the edge evaluation takes its maximum value for γ=0.8. Even though we have chosen γ=0.8, two remarks should be made: Firstly, values of γ from 0.6 to 0.9 produce similar results. Secondly, γ can be adjusted depending on the relative seriousness of broken edges as against thickened edges, for a given application. In general, γ should be fairly high, since filling breaks in edges is usually a more difficult task than edge thinning.

To show that the peaks in Figure 8 are actually caused by more or less well formed edges, we give in Figure 10 an analogous plot, but for the test image of pure noise. The forms of the curves are quite different, without any well-defined peaks for the higher values of Y. However, this graph does reveal a noteworthy property of the edge evaluation measure: Even on an image of pure noise, it is possible to choose a threshold which gives a moderately high value of the edge evaluation measure. At first thought, this may seem to be a defect. However, on reflection, it is clear that this is an inevitable characteristic of any such measure based on local good form. Because of overlap between the neighborhoods to which the edge operators are applied, an isolated noise point will produce a correlated set of edge pixels. For example, the three-level operator will produce a tiny ring. Even though this ring is highly curved, it is coherent, and will receive a moderate edge evaluation score. This evaluation score for isolated noise spots can be computed analytically as an intrinsic property of each edge detection scheme. For an image of pure noise, as used for Figure 10, the evaluation is somewhat lower than one would expect, apparently because of interference between adjacent noise pixels. In summary: even in a noisy image, there will be a certain occurence of well-formed edges, either by accidental alignment or as an artefact of the edge detection scheme used. It is not the fault of the edge evaluation measure that it reflects this unavoidable property of the images and detection schemes used.

Figure 11 illustrates the effects of various levels of noise on the edge evaluation measure. For clarity, only a subrange of the data is plotted. Outside this subrange the plots for the different noise levels tend to converge. The results show a consistent pattern: The peaks decrease in step with the signal-to-noise ratio. Below SNR=10, there are no clear peaks, but the shapes of the curves show that the presence of edges still has some effect on the edge evaluation. Although we have not pursued the matter, this suggests that an edge evaluation measure might be based on the value E of the measure for the given image relative to the value measured for the same detector on an image of pure noise. But such a relative measure would be useful only for cases of high noise, when E has no clear peaks, and would not be a good means of comparing the outputs of different edge operators. Away from the peaks, the evaluations for the different noise levels tend to become similar, while retaining the same ordering. This shows that a poor threshold leads to a bad selection of edges, no matter how noisy the original image.

All the above results are pretty much what one would expect intuitively from a measure of edge quality. They thus serve to confirm the validity of the edge evaluation measure. While the figures show the results for the rings image, the results for the vertical image are similar, and if anything, more distinct, since a vertical edge can be more cleanly digitized, and has not even the slightest curvature.

Comparison of Detectors

Having established that the measure E behaves well, we now present a comparison among the ten edge enhancement operators mentioned above. Every operator was applied to the test image at the seven different signal-to-noise ratios, and at each noise level the threshold was adjusted to maximize E. Figures 12 and 13 show these maximum values for the differential and template-matching operators respectively using the rings image. As expected, these results show that the three by three operators are far better than the two by two operators at detecting edges in the presence of noise. Among the three by three operators, the three-level operator is clearly the best, and the compass gradient the worst. The other four operators produce results of about the same quality. The same ordering is preserved if we subtract out the intrinsic response for each operator on pure noise, although the separations are not so great.

Analogous results for the vertical edge image are shown in Figures 14 and 15. They are not directly comparable, especially at the lower SNRs, because the rings image has a greater density of edges. However, some general remarks can be made. Firstly, as explained earlier, the vertical edge gives a higher evaluation. Secondly, the evaluations of the four three by three differential operators are more spread out. This can be attributed to relative orientation biases in the four operators which are brought out by the vertical edge, but which are cancelled out over the full range of edge orientations in the rings image.

Overall, this comparison is in accord with the findings of Abdou and Pratt. Our results differ from theirs only when the difference between operators is small by both measures. They also find the three by three operators consistently better than the two by two. However, at the highest signal to noise ratios, the performance of the two by two operators, according to their figure of merit, approaches that of the three by three, while our measure still reveals a considerable difference. This shows that while the two by two operators can properly locate edges at low noise levels, they poorly estimate the edge direction.

By both their measure and ours, the compass gradient is the worst of the three by three operators, but their figure of merit, while rating the three-level operator fairly highly, does not show it as clearly superior in all cases. These small discrepancies are not at all surprising, since the two edge evaluation schemes, after all, measure quite different characteristics of edges. The general agreement between the two schemes is encouraging: It seems both to confirm, in large part, the edge operator ratings of Abdou and Pratt, but from a different perspective; and also to strengthen our confidence in the usefulness of the measure E.

Effects of Preprocessing

Quite a number of techniques have been proposed for improving the quality of edge detection. We present here some experiments to demonstrate how the effect of a selection of these techniques is reflected in the edge evaluation measure E.

For coping with the effects of noise, two commonly used techniques are mean and median filtering, that is, each pixel in the original image is replaced by respectively the mean or median of the grey levels in a neighborhood around the pixel. This has the effect of smoothing out irregularities due to noise. However, as is widely known, mean filtering has the unfortunate side effect of blurring or thickening real edges, so median filtering is often preferred since it does not suffer from this defect. On the other hand, thickening of edges can usually be dealt with by non-maximum suppression on the edge gradient magnitudes - that is, a pixel has its magnitude set to zero unless it is a local maximum among those pixels which lie closest to the edge gradient direction.

Figure 16 shows the effects of mean and median filtering on E for different neighborhood sizes. As can be seen, the edge quality as measured by E is improved by both mean and median filtering, but if the neighborhood is too large, mean filtering causes a decrease in edge quality because of blurring, while mean filtering suffers from no such defect, although it seems less effective with smaller neighborhoods. This graph also shows the effect of applying non-maximum suppression to the edge magnitude output after mean filtering. Even when no averaging is done (the case of a one by one neighborhood), non-maximum suppression causes a small improvement in edge quality, by counteracting the slight blurring introduced by the edge operator masks. When the averaging is done over a larger neighborhood, the improvement is more significant, reaching a maximum when the mean filtering is done over the same sized neighborhood as the non-maximum suppression (that is, three by three).

That the above interpretation of Figure 16 is correct is shown in Figures 17 and 18, which are analogous, but use $\gamma=0.6$, giving more weight to edge thinness, and $\gamma=1.0$, showing the effect on the continuation measure alone. These graphs reveal the relative effects of the operators on edge continuity and edge thinness.

Peleg [1978] has devised a technique for edge improvement that fills small gaps and straightens out irregularities in edges. The effect of this process on edge output is presented in Table 2. While Peleg's technique certainly improves the form of edges, it has the undesirable side-effect of thickening them. However, this can be overcome by applying non-maximum suppression, as is also shown in Table 2. Again, the relative effects of this process on edge continuation and edge thinness can be seen by comparing the results for the different values of γ.

CONCLUSIONS

We have presented a method for evaluating the quality of edge detector output based solely on the local good form of the detected edges. It combines two desiderata of a well-formed edge - good continuation and thinness. This measure behaves as one would like under the effects of change of threshold, noise, blurring and other operations. The comparison experiments show that the results obtained with this measure are similar to those obtained with a measure based on the discrepancy of the detected edge from a known actual edge position. The small differences between the two methods reveal some properties of the operators not brought out by the other approach.

Like other evaluation measures, ours can be used to compare the effectiveness of different edge detection schemes and edge improvement schemes on synthetic images. However, since our measure does not require knowledge of the true location of edges, it has much wider application. It can be used to adjust parameters, such as thresholds, for optimum detection of edges in real images for which edge location is unknown. The evaluation of the detected edges can also serve as an indication of the quality of the original image. Further, the approach of using local coherence can be extended to the evaluation of other local feature detectors.

REFERENCES

1. Abdou, I. E. (1978) "Quantitative methods of edge detection," USCIPI Report 630, Image Processing Institute, University of Southern California, Los Angeles, CA 90007.

2. Abdou, I. E. & Pratt, W. K. (1979) "Quantitative design and evaluation of enhancement/thresholding edge detectors," Proc. IEEE, Vol. 67, No. 5, pp. 753-76[illegible].

3. Bryant, D. J. & Bouldin, D. W. (1979) "Evaluation of edge operators using relative and absolute grading," PRIP 1979, pp. 138-145.

4. Cornsweet, T. N. (1970) Visual Perception, Academic Press, New York, pp. 268-284.

5. Dember, W. N. (1966) Psychology of Perception, Holt, Rinehart & Winston, New York, pp. 196-206.

6. Fram, J. R. & Deutsch, E. S. (1974) "A quantitative study of the orientation bias of some edge detector schemes," Computer Science Tech. Report 285, University of Maryland, College Park, MD 20742.

7. Fram, J. R. & Deutsch, E. S. (1975) "On the quantitative evaluation of edge detection schemes and their comparison with human performance," IEEE Trans. Comp., Vol. C-24, No. 6, pp. 616-628.

8. Gregory R. L. (1974) Concepts and Mechanisms of Perception, Charles Scribner's Sons, New York, pp. xxx-xl.

9. Peleg, S. (1978) "Straight edge enhancement and mapping," Computer Science Tech. Report 694, University of Maryland, College Park, MD 20742.

10. Peleg, S. & Rosenfeld A. (1977) "Determining compatibility coefficients for relaxation processes," IEEE Trans. Systems, Man & Cyber., Vol. SMC-8, No. 7, pp. 548-555.

11. Pratt, W. K. (1978) Digital Image Processing, John Wiley & Sons, New York, pp. 495-501.

12. Weszka, J. S. & Rosenfeld, A. (1978) "Threshold evaluation techniques," IEEE Trans. Systems, Man & Cyber., Vol. SMC-8, No. 8, pp. 622-629.

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged, as is the help of Sherry Palmer in preparing this paper.

γ =	0.0	0.2	0.4	0.6	0.8	1.0
Maximum value	1.000	0.890	0.856	0.852	0.876	0.922
At threshold (% of range)	88-98	74	68	60	56	46

Table 1. Rings image, SNR=10, three level operator. Maximum value of edge evaluation measure, and threshold at which this occurs, for various values of γ.

γ =	0.6	0.8	1.0
SNR 10	0.771	0.759	0.757
Enhanced	0.786	0.790	0.806
Enhanced & non-maximum suppression	0.841	0.823	0.805

Table 2. Effect of Peleg's edge enhancement procedure on edge evaluation.

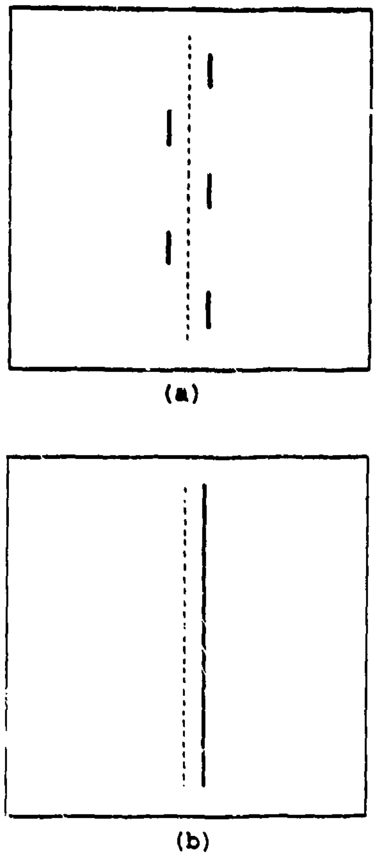


Figure 1. (a) Disconnected edge, and (b) well-connected edge, both with equal displacement from ideal edge position. (Ideal edge position shown by dotted line, detected edge by heavy line.)

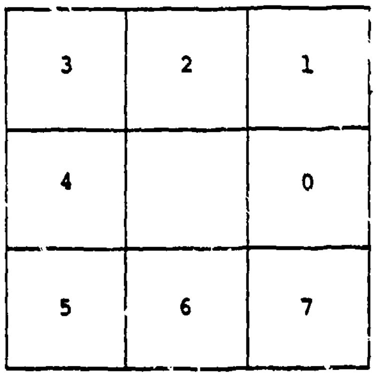


Figure 2. Numbering system for neighbors.

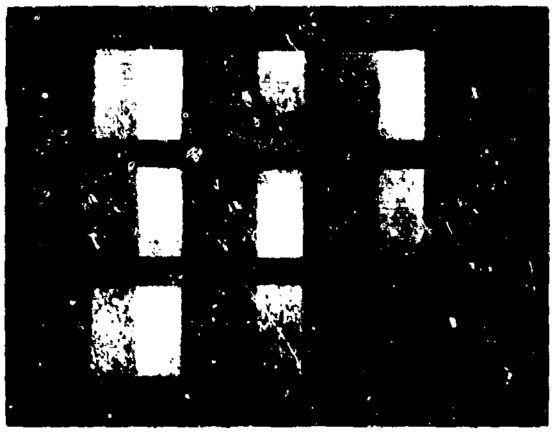


Figure 3. Vertical edge test image, with various levels of noise. From left to right, top to bottom: no noise, SNR = 100, 50, 20, 10, 5, 2, and 1. (Note: these images are at twice the scale of those in Figure 4.)

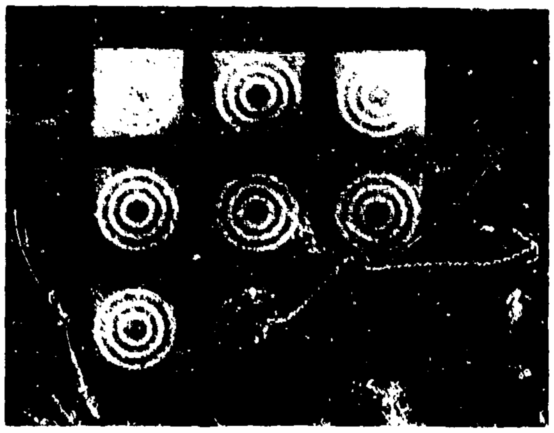


Figure 4. Rings test image, with various levels of noise. From left to right, top to bottom: no noise, SNR = 100, 50, 20, 10, 5, 2 and 1.

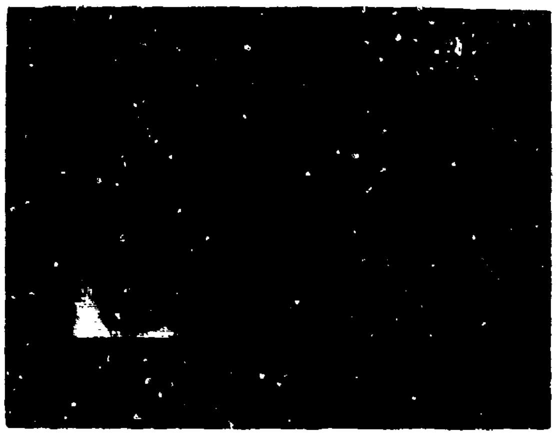


Figure 5. Histogram of edge magnitude obtained by applying three-level operator to rings image at SNR 50.

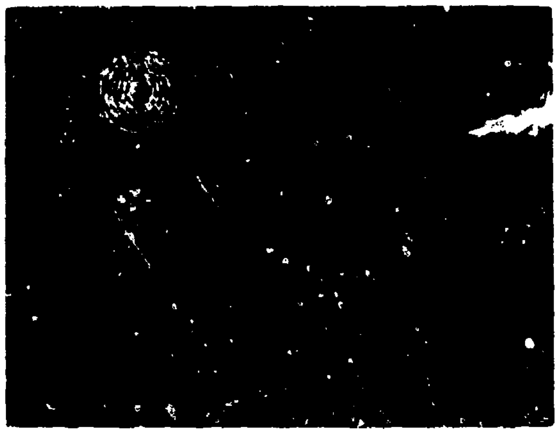


Figure 6. Edge pixels extracted by thresholding edge magnitude (three-level operator) on rings image at SNR 50. Thresholds, from left to right, top to bottom: 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80% and 90% of range.

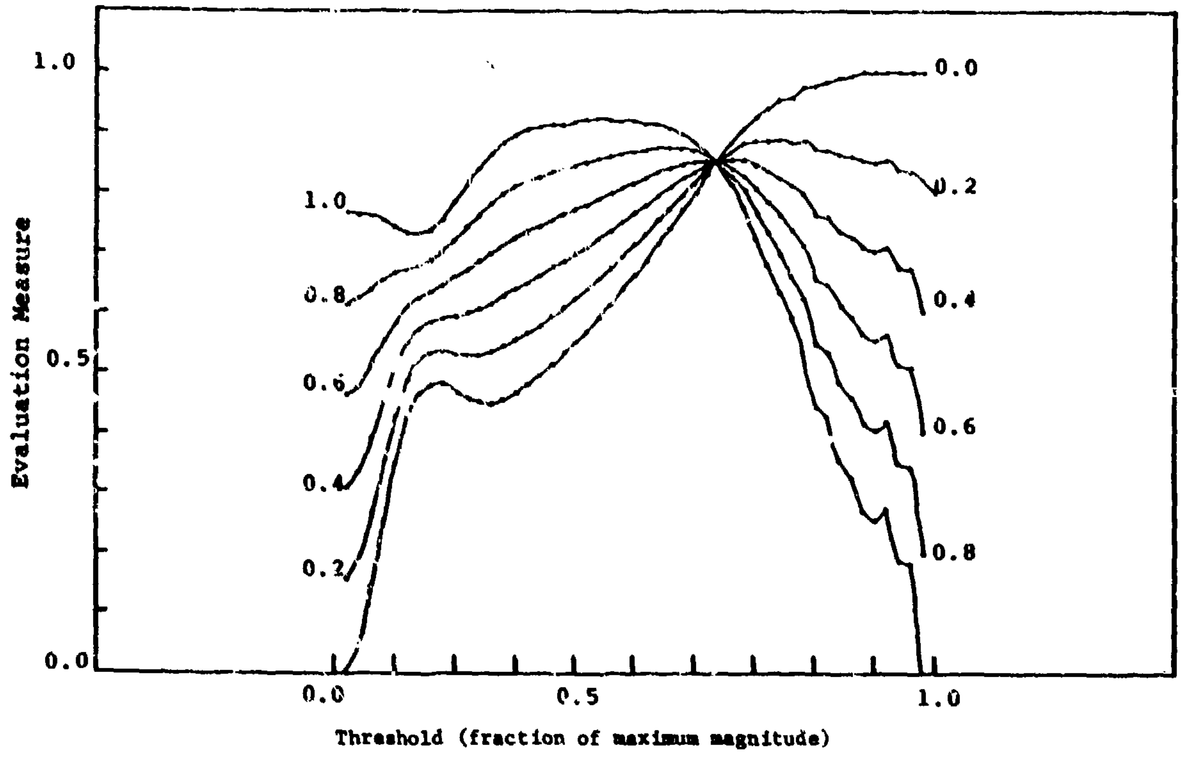


Figure 7. Using rings test image at SNR 50 and three-level operator: edge evaluation against threshold for various values of parameter γ.

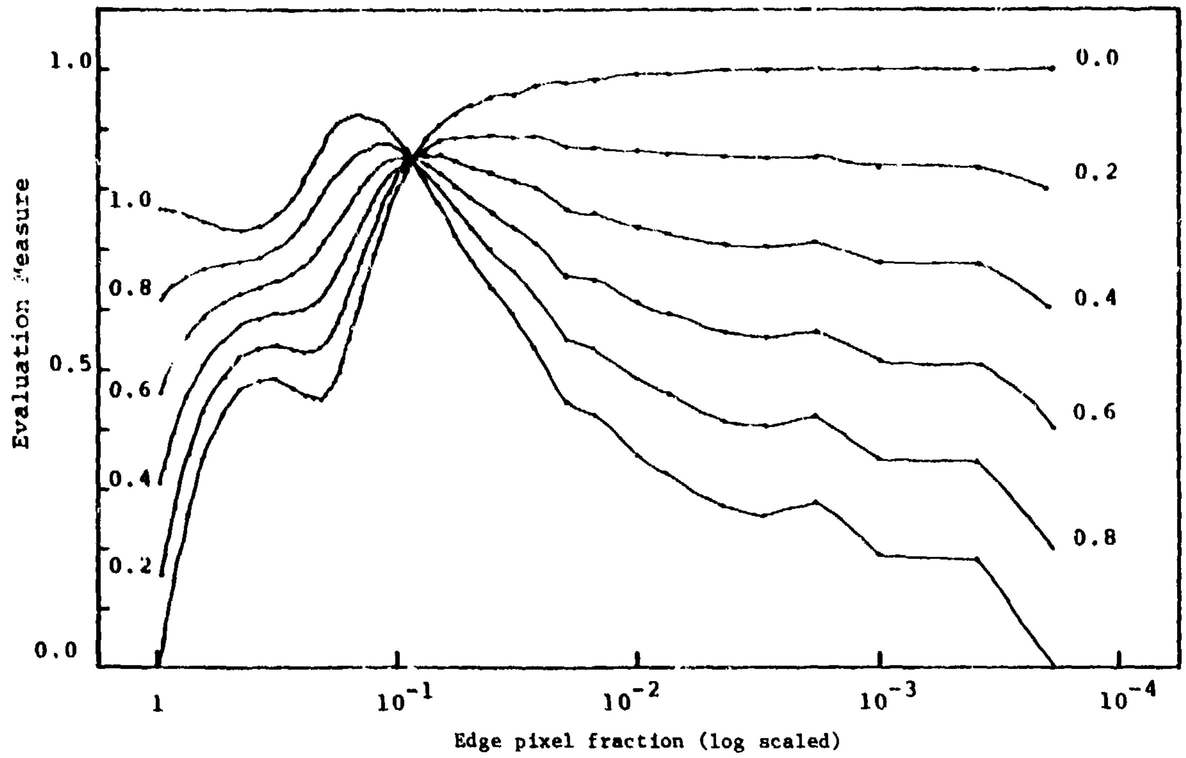


Figure 8. Using rings test image at SNR 50 and three-level operator: edge evaluation against fraction of edge pixels at each threshold, for various values of parameter γ.

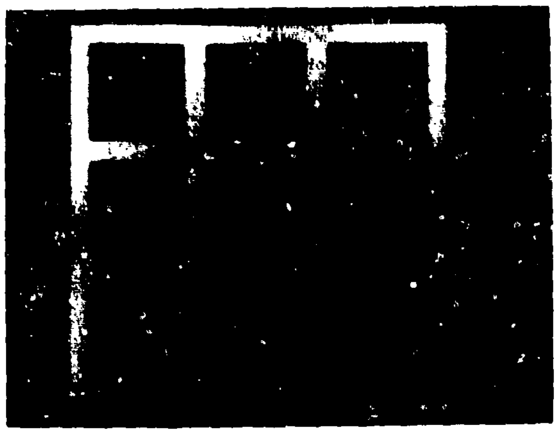


Figure 9. Edge pixels extracted by thresholding edge magnitude from three-level operator on rings image at SNR 50. Thresholds from left to right, top to bottom: 48%, 50%, 52%, 54%, 56%, 58%, 60%, 52%, 64% of range.

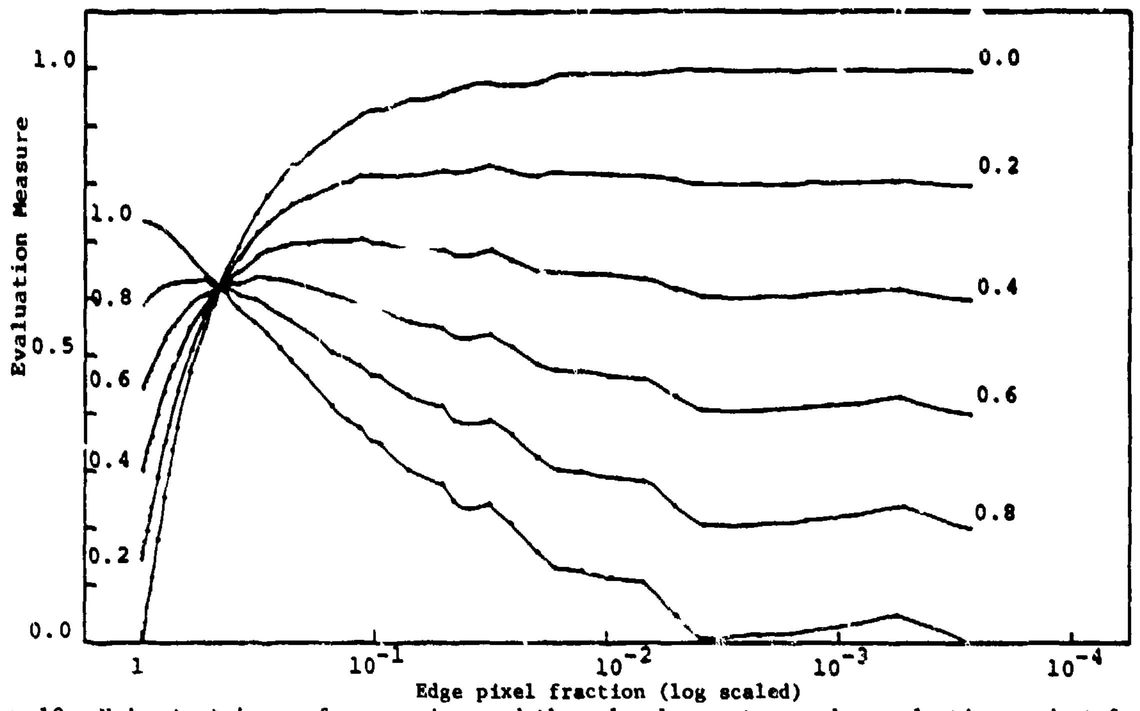


Figure 10. Using test image of pure noise, and three level operator: edge evaluation against fraction of edge pixels, for various values of parameter γ.

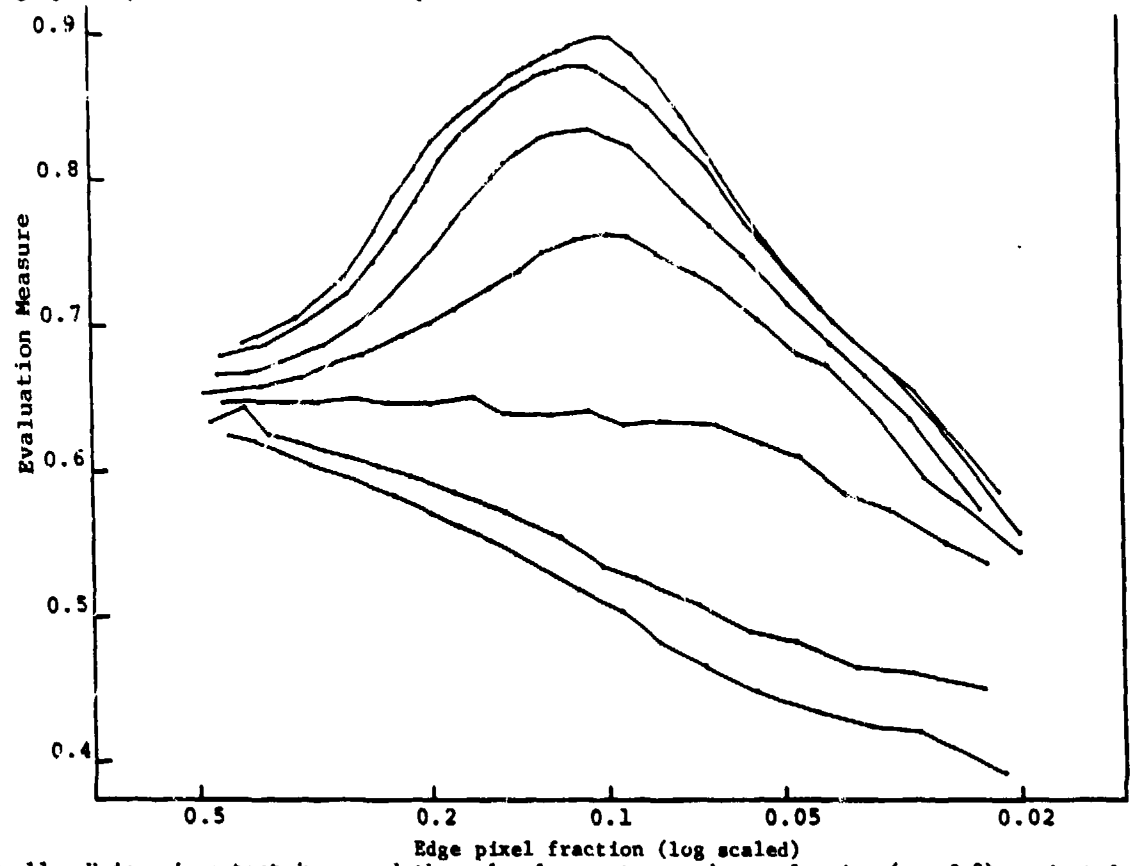


Figure 11. Using rings test image and three-level operator: edge evaluation (γ = 0.8) against fraction of edge pixels at each threshold, for various values of SNR (top to bottom curve: 100, 50, 20, 10, 5, 2, 1).

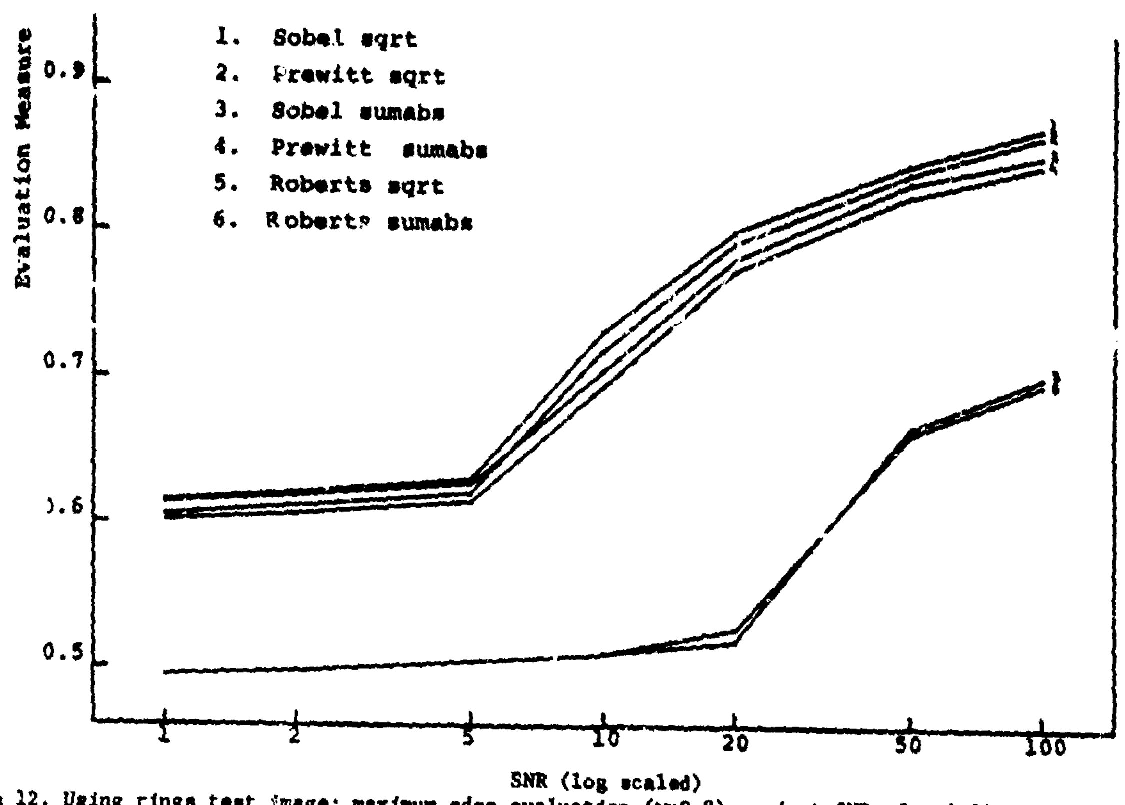


Figure 12. Using rings test image: maximum edge evaluation (γ=0.8) against SNR, for differential operators.

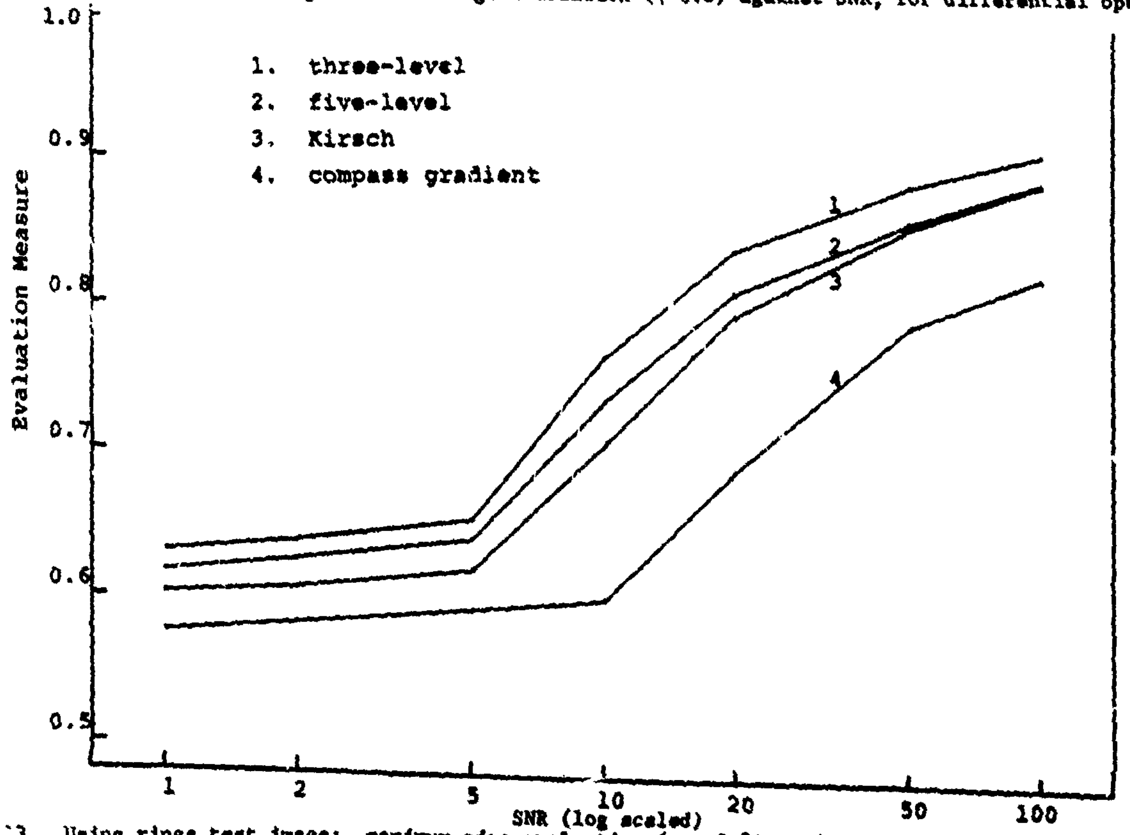


Figure 13. Using rings test image: maximum edge evaluation (γ = 0.8) against SNR, for template matching operators.

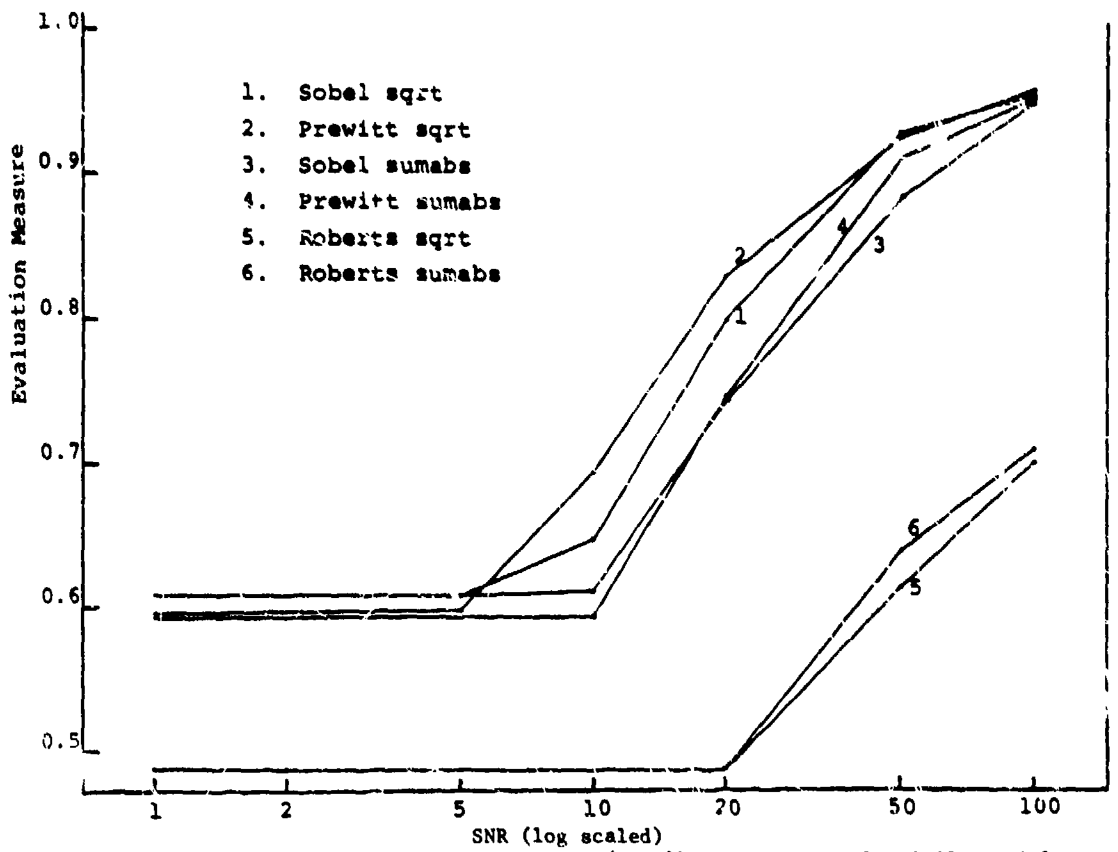


Figure 14. Using rings test image: maximum edge evaluation (γ=0.8) against SNR, for differential operators.

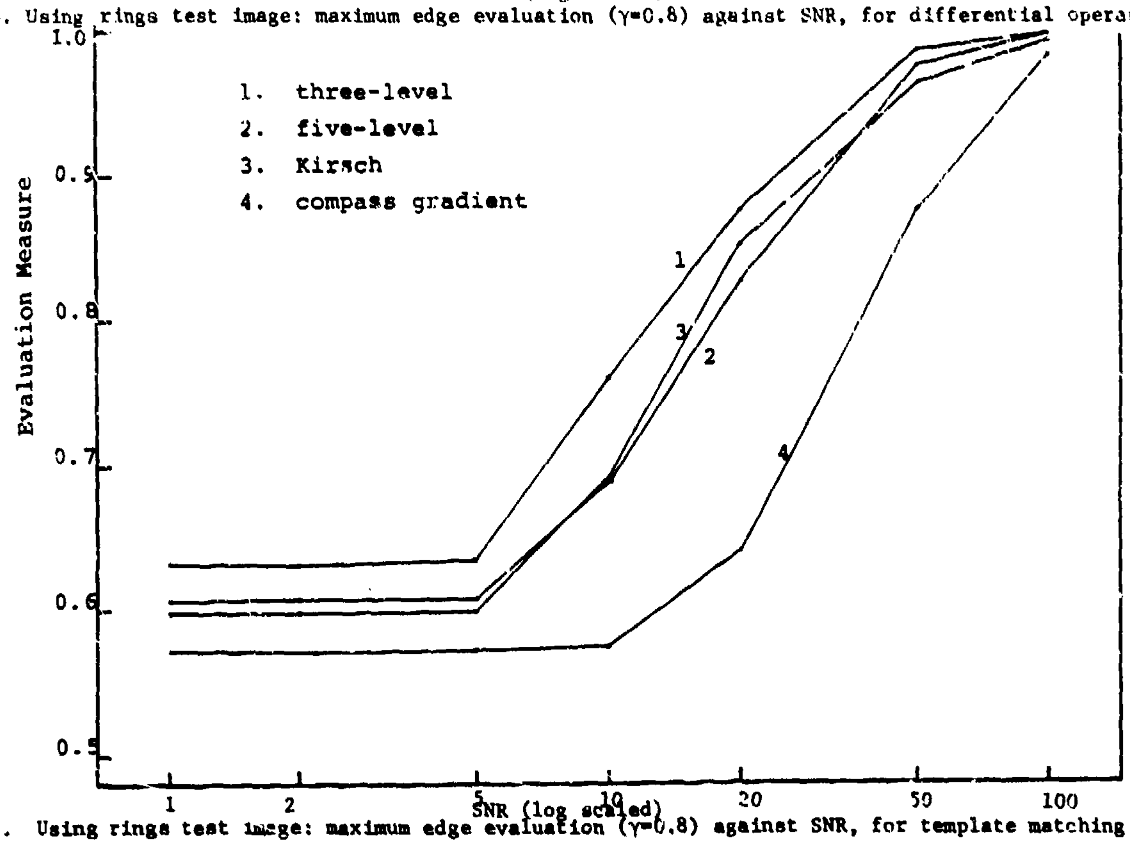


Figure 15. Using rings test image: maximum edge evaluation (γ=0.8) against SNR, for template matching operators.

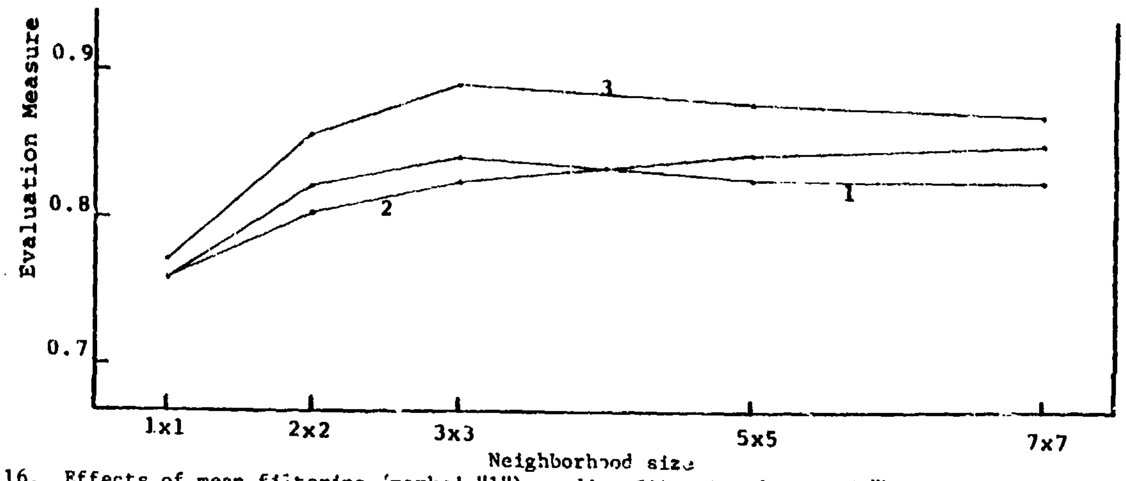


Figure 16. Effects of mean filtering (marked "1"), median filtering (marked "2"), and mean filtering followed by non-maximum suppression (marked "3"), against size of neighborhood (Uses rings test image at SNR 10, three-level operator. and γ = 0.8.)

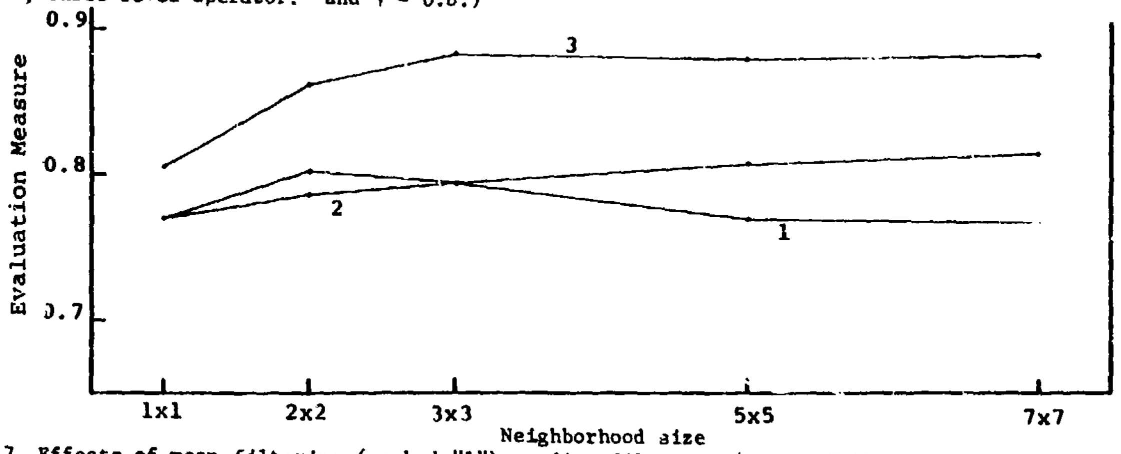


Figure 17. Effects of mean filtering (marked "1"), median filtering (marked "2"), and mean filtering followed by non-maximum suppression (marked "3"), against size of neighborhood. (Uses rings test image at SNR 10, three-level operator, and γ = 0.6.)

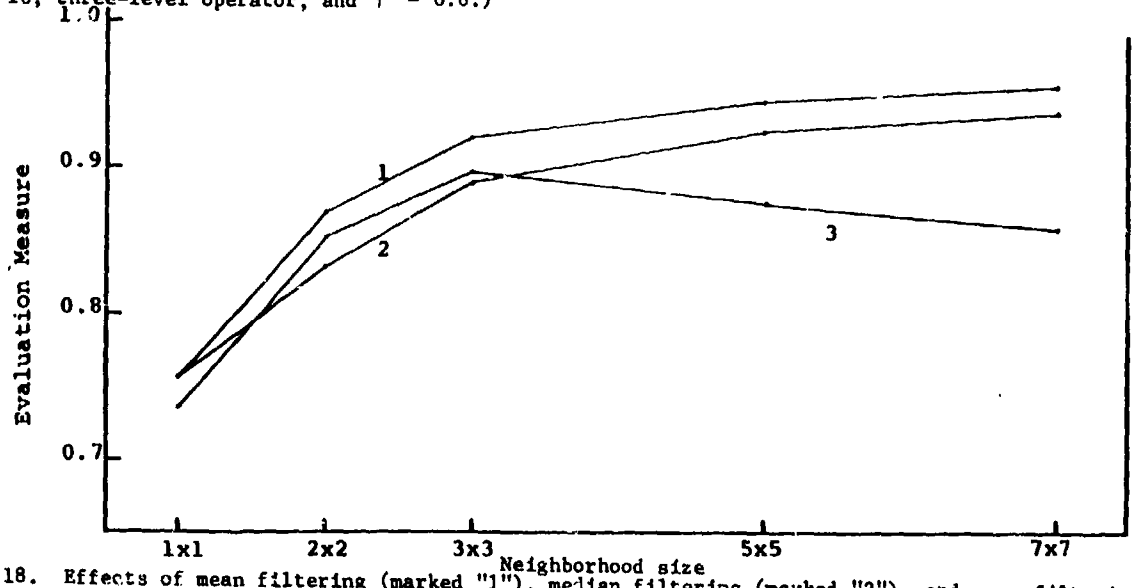


Figure 18. Effects of mean filtering (marked "1"), median filtering (marked "2"), and mean filtering followed by non-maximum suppression (marked "3"), against size of neighborhood. (Uses rings test image at SNR 10, three-level operator, and γ = 1.0.)

TOWARDS A REAL TIME IMPLEMENTATION OF THE MARR AND POGGIO STEREO MATCHER

H. K. Nishihara and N. G. Larson
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

INTRODUCTION

This paper reports on research--primarily at Marr and Poggio's [9] mechanism level—to design a practical hardware stereo-matcher and on the interaction, this study has had with our understanding of the problem at the computational theory and algorithm levels. The stereo-matching algorithm proposed by Marr and Poggio [10] and implemented by Grimson and Marr [3] is consistent with what is presently known about human stereo vision [2]. Their research has been concerned with understanding the principles underlying the stereo-matching problem. Our objective has been to produce a stereo-matcher that operates reliably at near real time rates as a tool to facilitate further research in vision and for possible application in robotics and stereo-photogrammetry. At present the design and construction of the camera and convolution modules of this project have been completed and the design of the zero-crossing and matching modules is progressing. The remainder of this section provides a brief description of the Marr and Poggio stereo algorithm. We then discuss our general approach and some of the issues that have come up concerning the design of the individual modules.

There are two distinct approaches to identifying correspondences between locations in the left and right images of a stereo pair. The first is to focus on the local pattern or arrangement of some fine-scale matching primitive, attempting to determine the mapping between left and right images which best correlates these patterns [cf 5, 6, 1, 4, 8, 12]. The other approach [10] is to focus on the use of primitives sensitive to image details at different scales so that matching can be accomplished first at the coarsest scale and then at successively finer scales. The density of such primitives in an image is tied to the scale at which they are sensitive which makes it possible to use a simple matching rule such as:

For each primitive element in the left image, look in a horizontal interval in the right image about the corresponding location. If there is only one primitive there that could match it, accept it as the matching element. If there are no potential matches in the search interval, note this fact as evidence that the search window may be improperly positioned in the right image. If there is more than one potential match in the search interval, the match is ambiguous so skip over this point.

The length of the horizontal search interval can be chosen so that a sufficient percentage of unique matches is found. The coarse-scale primitives allow the use of a larger search interval, thus gaining disparity range in exchange for resolution and the density of matches that can be obtained. Matching primitives at a finer scale—which requires a shorter search interval—can then be accomplished by positioning the search interval using the rough disparity information obtained from matching the coarser primitives.

Marr and Poggio [10] found that, for this second approach, peaks in the rate of intensity change in the image at a given scale of resolution were the appropriate type of matching primitive. Peaks in the rate of intensity change along the direction of the local intensity gradient—or equivalently signed zero-crossings in the second derivative—correlate with physical markings and edges on surfaces. However, zero-crossings in the second derivative of the image are most sensitive to details at the finest scale of resolution. This problem can be dealt with by first low pass filtering the image to attenuate high spatial frequency structure above the scale of resolution desired. It turns out that Gaussian smoothing offers the best compromise for attenuating fine scale structure in the image while preserving the local geometry at larger scales [7]. Marr and Hildreth [7] found that with weak restrictions, zeros in the Laplacian of a Gaussian convolved image—$\nabla^2(G*I)$— gave the desired result. Furthermore, $\nabla^2(G*I)=(\nabla^2G)*I$ and ∇^2G can be closely approximated by the difference of two Gaussians having different space constants but normalized to have the same volume.

DESIGN

The hardware implementation project grew out of our effort in spring 1980 to construct supporting hardware for a VLSI convolution chip being developed at that time at Hughes Research Labs [11]. A "serpentine memory" device was required to buffer video output from a TV camera so that 26 successive image lines could be fed to the convolver in parallel at video rates. The VLSI circuit would then convolve the image with a built-in 26 by 26 difference of Gaussian mask, producing a video convolution signal—lagging the TV camera input by 26 lines. In the process of doing this, we found that a practical Gaussian convolver could be designed using standard digital TTL technology. This approach took advantage of the Gaussian's separability which makes it possible to decompose the two-dimensional convolution into two successive one-dimensional Gaussian convolutions, reducing the number of multiplications required at each image point from m^2 to $2m$ where m is the mask diameter. Although the speed of such a device would be slower—1 MHz—than the promised speed of the Hughes chip, it had several advantages. The extensive analog circuitry necessary to operate the Hughes chip could be avoided; the all digital design would be easier to debug and would allow more precision; variable sized Gaussians would be allowed; and we would have better logistical control over the project's time frame. Shortly thereafter we also worked out a rough design for a stereo-matcher using the same technology.

During this period the following guidelines for our design were formulated:

(1) *The stereo-matching system should be raster-based and pipelined.* Our objective here was to eliminate large "frame buffer" memories for storing intermediate results during the operation of the stereo-matcher where they were unnecessary. We already had a design for doing convolutions that operated directly off of a TV raster input and it also seemed that the stereo-matching could be accomplished directly off of parallel raster inputs from the left and right camera-convolver combinations. This freedom from frame buffer requirements allows us to design for image sizes much larger than would otherwise be convenient to manage—the present system design handles images with up to 1024 pixels on a line and places no restriction on the number of lines in the image.

(2) *The system should be modularized in a way that facilitates development.* When we began we were quite certain about the kind of convolution that was needed for stereo-matching but we were less sure about the type of camera appropriate for the task or about many of the details of the matching hardware. So we wanted the overall system design to support at least two modes of operation. The full pipelined mode, illustrated in figure 1, which begins with two cameras and produces some form of raster disparity output as well as any of the intermediate raster signals such as the convolution values. The second mode which we call the memory to memory mode allows us to use an array stored in computer memory as the source of input to any of the modules, and another array as the destination for its output. This makes it possible to design and debug the modules individually using a host computer—in our case the MIT AI Laboratory's Lisp machine—as a sophisticated test bed. Figure 2 shows the way the convolution module is interfaced to the Lisp machine's Xbus to accomplish this. Variations on this mode allow the whole system to be run—at slower rates—incorporating experimental designs for one or more of the modules as microcoded simulations on the Lisp machine.

(3) *The system should be developed in a technology that provides good design aids and allows rapid construction and easy circuit modification.* The best hardware development support at our lab is for TTL technology on machine wire wrapped boards. Wire wrapped circuits are also easy to modify during the debugging process. We feel that this gives us the best combination of computational power, development ease, and circuit reliability. We expect that once experience has been gained with the prototype system, a transfer to other technologies such as VLSI will become be more practical.

CAMERAS

Our prototype system requires two synchronized raster inputs from left and right cameras at a pixel rate of about 1 MHz. The sources can be cameras or images stored in computer memory. A particular concern for stereo-matching is that geometric distortion between left and right images be held to a minimum. For the memory to memory mode of operation, this can be accomplished by using the same image digitizer for both images. For real time applications, however, the geometric distortion of the cameras should be less than a pixel diameter if possible. There are means for compensating for fixed geometric distortions between two cameras, but it would be desirable to avoid having to use them.

We considered three types of camera for real time operation, beam deflection devices such as vidicons, two-dimensional CCD cameras, and one-dimensional CCD cameras with mirror deflection systems. The vidicon family of cameras is the best developed, and a wide range of devices is available. However, they generally have large—on the order of one percent or more—geometric distortions and for the most part are designed to operate at speeds higher than 1MHz.

The two-dimensional CCD arrays have high geometric accuracy, but at present they have limited image sizes and tend to have pixel, line or column flaws. Otherwise, these devices have good light sensitivity, they are small, and at least some versions can be run at the speeds we desire. We expect that in the long run these will be the most practicable cameras, especially for robotics applications where more rapid frame rates maybe preferable to very large image dimensions—given the constraint of a fixed pixel rate.

A one-dimensional solid state CCD array camera in conjunction with a scanning mirror provides good geometric precision with large image dimensions and no pixel flaws. The only shortcoming of this type of camera for our purposes is its lower light sensitivity. This problem occurs because the integration time of the individual sensors is limited to the time spent on a single line of the image—for a 1024 element array running at 1MHz this is about 1ms. Two-dimensional sensors in comparison have integration times limited by the time taken to scan the whole frame. During the summer of 1980, we acquired a 256 element linear array camera constructed by R. Bishop as a thesis project at MIT. When operated at a 1MHz pixel rate, this camera had an integration time of approximately 0.25ms and performed reasonably well under studio lighting conditions.

From this we decided that an in house designed 1024 pixel linear array scanner would best meet our requirements over the next year or so. The camera we developed—see figure 3—makes heavy use of off-the-shelf components. A Reticon evaluation board is used to operate the linear array and most of the necessary analog circuitry is provided on that circuit board. A separate analog to digital converter was designed and a third circuit board was designed to generate the ramp signal for the mirror controller as well as the vertical and horizontal syncs. A controller board and closed-loop galvanometer produced by General Scanner Inc. is used to move the mirror. The mirror sweep is specified to be linear to within 0.15 percent of the peak to peak sweep angle and mirror repeatability is claimed to be within 1 second of arc.

Our preliminary performance observations are that the camera's geometric precision and repeatability meet our requirements. The camera's light sensitivity also seems satisfactory and we hope to be able to operate the camera-convolver combination at normal office light levels. We have had difficulty adjusting the balance controls on the Reticon board—for the even and odd pixel signals which are output separately by the linear array. While this problem is not significant when the camera output is convolved with a difference of Gaussian mask, we may improve the analog circuit or to do digital compensation to support other uses of the camera. More quantitative performance measurements are in progress.

The overall simplicity of this design makes it easy for us to tailor the cameras to changing requirements that may develop as the overall project evolves. For example, should it become necessary to vary the relative vertical alignment between the left and right cameras as the scan progresses across the image—to compensate for changes in surface depth—the ramp generator can be modified easily to accept a dynamic vertical offset correction signal from the stereo-matcher.

An effort has also been made to standardize the interface to the convolver so that other camera types can be used in the future with a minimum of difficulty. The interface requires (1) a pixel clock signal with a frequency of up to 1MHz, (2) vertical sync, (3) horizontal sync, and (4) 8 bit pixel data. The convolver accepts camera line lengths from 1 to 1024 pixels.

CONVOLVER

The convolution module was the central focus of the first half of our development effort because of the large computational requirements that arise in its operation. For digital Gaussian convolution with a 32 by 32 mask size, a minimum of 32 multiplies are required for each pixel of the image. To maintain adequate precision, the first one-dimensional convolution requires 16 8x8 multiplies while the second one-dimensional convolution requires 16 8x16 multiplies. Using TRW multiplier chips we are able to achieve a pixel rate of just under 1MHz. Higher pixel rates should be obtainable with more parallel or analog designs such as the Hughes convolver chip.

Another issue is whether to compute a difference of two Gaussians or the Laplacian of a single Gaussian convolution. Generally, given Gaussian convolutions with a given precision, the difference of Gaussian approach offers a better signal to noise ratio because of the second-order differences involved in computing the Laplacian. The design of two copies of the same convolution circuit would also be easier so the difference of Gaussian was selected.

The resulting hardware takes 8 bit pixels as input image data and produces signed 16 bit numbers as output. The 32x32 Gaussian mask size allows difference of Gaussian convolution masks with central positive diameters *w* of from 2 to about 12 pixels. In memory to memory mode on the Lisp machine, a 1000x1000 by 8 bit image can be convolved and the result stored as a 1000x1000 by 16 bit array in about 20 seconds—1.5 seconds for convolving and the remainder for paging between disk storage and memory.

With the completion of the linear array camera, we have been working with a camera-to-convolver-to-display mode using a microcode loop to read the convolver output and write the sign bits of the convolution to the bit map of the Lisp machine's black and white monitor. This gives us a real time display of zero-crossings from the camera and we have begun to use it to get a better familiarity with what the world looks like in these terms. One quickly notices that zero-crossings occur locally in response to the features with the largest contrast and they tend to form closed loops with diameters on the order of w. If there are sharp intensity edges the zero-crossings follow them locally and lower contrast or smaller features have little influence. In the absence of more significant intensity fluctuations, however, even very small intensity variations such as are on a sheet of white bond paper give rise to zero-crossings. The zero-crossings generated by lower contrast features are more susceptible to noise such as the variations due to 120 cycle illumination flicker. This particular noise source should not be a problem for our matching scheme because the left and right cameras will be synchronized and so will see the same zero-crossing distortions. They may even contribute to the texture on otherwise uniform surfaces.

MATCHING

The matching module presently being designed will have three components, (1) a zero-crossing coder which detects zero-crossings in the convolution raster signal and codes the orientation, and possibly the gradient or curvature of the zero-crossing contour at that location; (2) a matcher that takes the coded zero-crossing rasters from the left and right cameras and computes the disparities of unique matches when they occur as well as information about the absence of any matches at locations where they were expected; and (3) some form of statistics checker for filtering good matches from random incorrect matches which occur when the matcher's search window is improperly positioned.

The stereo matching algorithm developed by Grimson from Marr and Poggio's theory operates on roughly vertical zero-crossings in difference of Gaussian filtered images. For each such zero-crossing point in the left image, a window in the other image one line high and w in length (where w is the diameter of the positive part of the DOG filter) is searched. If there is only one zero-crossing in the window that matches the zero-crossing from the left image, it is taken as a candidate match to be further validated by a subsequent statistical check. Grimson has made effective use of this scheme in furthering our understanding of the stereo matching problem and in demonstrating that the Marr and Poggio theory is consistent with the known psychophysical data on stereo vision.

A straightforward hardware design was worked out for a matcher following the general idea of Grimson's algorithm. However in tests of a software simulation of the design, we found that it was very sensitive to small vertical misalignments between the two images and decided to study the problem further. Grimson had observed this problem also and solved it by running the matching algorithm several times with different vertical offsets between the two images, collecting the good matches—as determined by the statistics checking—from each. A major objective of the hardware implementation, however, is speed so it would be preferable to make the matching less sensitive to small misalignments or to find a way to correct the misalignments prior to the matcher to reduce the number of passes required over the image.

The matcher's sensitivity to small vertical misalignments has two roots: (1) matching is carried out pixel by pixel on roughly vertical zero-crossing segments that are often less than 5 pixels high in the image; and (2) the search window is along a single line. As a consequence, if there is a vertical misalignment of n lines between the left and right images, n points on one end of each vertical zero-crossing segment in the left image cannot match with the correct segment in the right image. This reduces the portion of zero-crossings finding matches which reduces the likelihood that the statistics checking will pass any of the matches. Worse, in cases where the misalignment is small enough to allow most candidate matches through, it becomes possible for zero-crossing points at the ends of vertical segments which do not "see" the correct segment in the other image to match to the next segment over and not be rejected by the statistics module. This later case produces sporatic mismatches with large disparity errors. One method—used by Grimson—for reducing this problem is to do the matching twice once driven from the left image and once from the right and then taking only those results found going both ways. This works, we think, because the accidental matches tend to occur differently in the two cases and so can be eliminated.

Vertical misalignments between the two cameras of a stereo system cannot be avoided. This is because the distances from a point in the field of view to the left and right cameras can be different giving rise to slightly different scales at that location in the images. This effect is most pronounced at the image's left and right edges and for larger vergence angles. Table 1 show some worse case estimates for the magnitude of this type of vertical misalignment for typical vergence angles and a 1000 line image with a 20 degree field of view. For example, with a vergence of 5 degrees—a target 11 feet away when the cameras are 1 foot apart—the expected vertical alignment of the cameras would be within plus or minus 7 lines or better.

There are several ways in which vertical offsets between the left and right images could be measured locally, so we are examining the possibility of a device for correcting misalignments at the zero-crossing detection stage of the system. Such a device has to be after the convolver if rapid changes in alignment are to be handled and it should be before the stereo-matcher. Our preliminary design would allow some generality in the type of zero-crossing information obtained as well as providing offset adjustment over an 8 to 16 line range.

DISCUSSION

We are beginning to experiment with the first part of our hardware system—the camera-convolver combination—with an emphasis on learning as much as we can about what this real time dimension can add to our understanding of the stereo matching problem. We are particularly interested in the degree to which the matcher and statistics modules can be streamlined by taking advantage of the repetitive nature of the real time system in conjunction with the image variations that can be expected between frames due to motion or illumination effects.

REFERENCES

1. Dev, P. Perception of depth surfaces in random-dot stereograms: a neural model. *Int. J. Man-Machine Stud.* 7, 511-528, (1975).

2. Grimson, W.E.L. A computer implementation of a theory of human stereo vision. Ph.D. thesis, Department of Mathematics, M.I.T. (1980). Also to appear in M.I.T. Press.

3. Grimson, W.E.L. and Marr, D. A computer implementation of a theory of human stereo vision. in *Proceedings of ARPA image understanding workshop*, ed L.S. Baumann, 41-45, (1979).

4. Hirai, Y. and Fukushima, K. An inference upon the neural network finding binocular correspondence. *Trans. IECE* J59-D, 133-140, (1976).

5. Julesz, B. Towards the automation of binocular depth perception (AUTOMAP-1). In *Proceedings of the IFIPS Congress*, ed C.M. Popplewell. Amsterdam: North Holland, (1963).

6. Julesz, B. Foundations of cyclopean perception (1971).

7. Marr, D and Hildreth, E. Theory of edge detection. *Proc. R. Soc. Lond.* B 207, 187-217, (1980).

8. Marr, D. and Poggio, T. Cooperative computation of stereo disparity. *Science* 194, 283-287, (1976).

9. Marr, D. and Poggio, T From understanding computation to understanding neural circuitry. *Neuralsciences Res. Prog. Bull.* 15 470-488, (1977).

10. Marr, D. and Poggio, T. A computational theory of human stereo vision. *Proc. R. Soc. Lond.* B 204, 301-328, (1979).

11. Nudd, G.R., Fouse, S.D. and Nussmeier, T.A. Development of custom-designed integrated circuits for image understanding. In *Proceedings of ARPA image understanding workshop*, ed L.S. Baumann, 1-9, (November 1979).

12. Sugie, N. and Suwa, M. A scheme for binocular depth perception suggested by neurophysiological evidence. *Biol. Cybernetics* 26, 1-15, (1977).

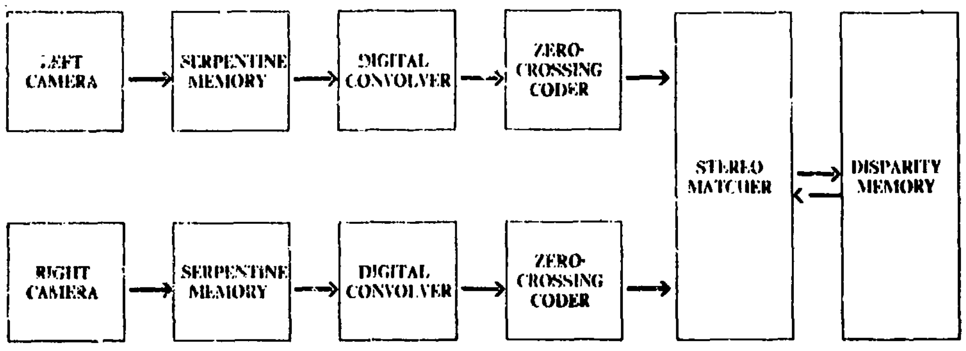


Figure 1. Modules of the hardware stereo-matcher. Heavy arrows indicate pipelined data paths.

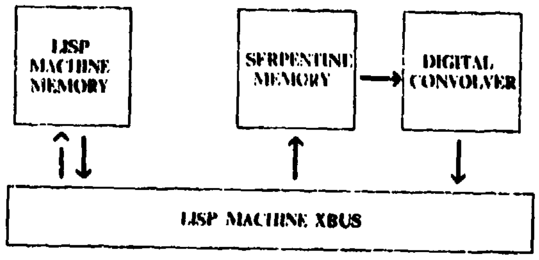


Figure 2. The serpentine memory and digital convolver combination on the Lisp machine Xbus. The Xbus is a 32 bit bus. Each module has its own data address—heavy arrows—and control address—thin arrows—on this bus. In memory to memory operation, a micro-code routine reads 4 bytes of image data from an array in memory, writes it to the serpentine memory's data address, and then reads 4 16 bit bytes from the convolver's data address and writes them to an output array in Lisp machine memory. This sequence is repeated until the entire input array has been scanned. This loop runs at about 1.5 μsec per pixel when no paging is required.

Table 1
Vertical Misalignment Magnitudes in a 1000 Line Image Due to Different Distances from Target to Cameras at 10 Degree Azimuth*

Vergence θ	Distance D/d	Ratio D_l/D_r	Bound on Misalignment (lines)
0	∞	1	± 0
1	56	1.0031	1
2	28	1.0062	3
3	19	1.0093	4
4	14	1.012	6
5	11	1.015	7
6	9.3	1.019	9
7	8.0	1.022	11
8	7.0	1.025	12
9	6.2	1.028	14
10	5.5	1.031	15

*θ is vergence angle in degrees.
D is mean distance from cameras to target at 10 degrees azimuth.
d is camera separation.
D_l and D_r are distances to target from left and right cameras.

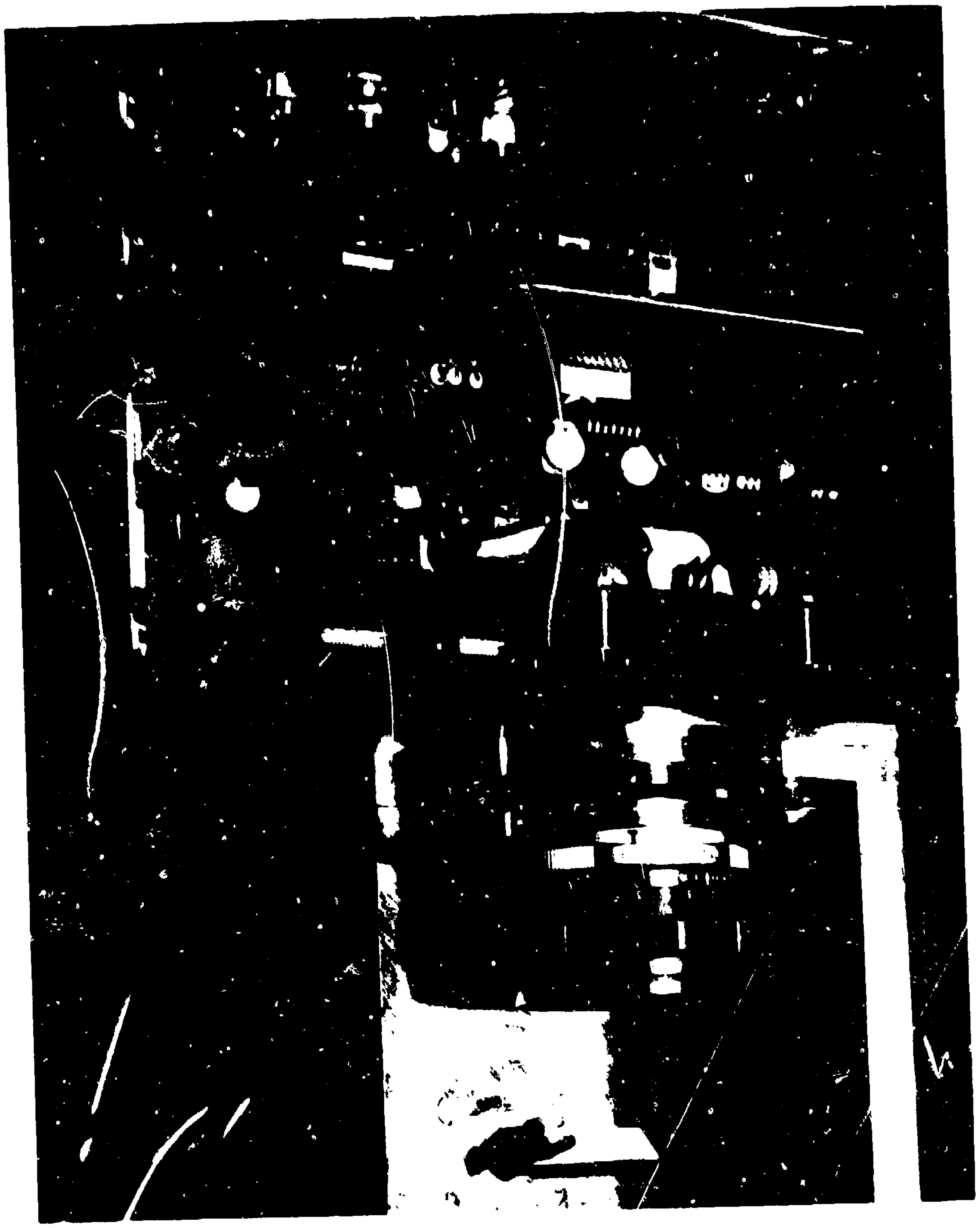


Figure 3. The linear array camera prototype. All circuitry for the camera other than the power supply is contained within the camera body--show here uncovered. Digital output is over a 56 foot ribbon cable.

An Iterative Image Registration Technique with an Application to Stereo Vision

Bruce D. Lucas
Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Image registration finds a variety of applications in computer vision. Unfortunately, traditional image registration techniques tend to be costly. We present a new image registration technique that makes use of the spatial intensity gradient of the images to find a good match using a type of Newton-Raphson iteration. Our technique is faster because it examines far fewer potential matches between the images than existing techniques. Furthermore, this registration technique can be generalized to handle rotation, scaling and shearing. We show how our technique can be adapted for use in a stereo vision system.

1. Introduction

Image registration finds a variety of applications in computer vision, such as image matching for stereo vision, pattern recognition, and motion analysis. Unfortunately, existing techniques for image registration tend to be costly. Moreover, they generally fail to deal with rotation or other distortions of the images.

In this paper we present a new image registration technique that uses spatial intensity gradient information to direct the search for the position that yields the best match. By taking more information about the images into account, this technique is able to find the best match between two images with far fewer comparisons of images than techniques which examine the possible positions of registration in some fixed order. Our technique takes advantage of the fact that in many applications the two images are already in approximate registration. This technique can be generalized to deal with arbitrary linear distortions of the image, including rotation. We then describe a stereo vision system that uses this registration technique, and suggest some further avenues for research toward making effective use of this method in stereo image understanding.

2. The registration problem

The translational image registration problem can be characterized as follows: We are given functions $F(x)$ and $G(x)$ which give the respective pixel values at each location x in two images, where x is a vector. We wish to find the disparity vector h which minimizes some measure of the difference between $F(x+h)$ and $G(x)$, for x in some region of interest R. (See Figure 1).

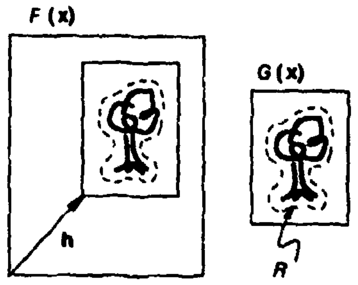


Figure 1: The image registration problem

Typical measures of the difference between $F(x+h)$ and $G(x)$ are:

- L_1 norm $= \sum_{x \in R} |F(x+h) - G(x)|$
- L_2 norm $= \left(\sum_{x \in R} [F(x+h) - G(x)]^2\right)^{1/2}$
- negative of normalized correlation

$$= \frac{-\sum_{x \in R} F(x+h)G(x)}{\left(\sum_{x \in R} F(x+h)^2\right)^{1/2}\left(\sum_{x \in R} G(x)^2\right)^{1/2}}$$

We will propose a more general measure of image difference, of which both the L_2 norm and the correlation are special cases. The L_1 norm is chiefly of interest as an inexpensive approximation to the L_2 norm.

3. Existing techniques

An obvious technique for registering two images is to calculate a measure of the difference between the images at all possible values of the disparity vector h—that is, to exhaustively search the space of possible values of h. This technique is very time consuming; if the size of the picture $G(\mathbf{x})$ is $N \times N$, and the region of possible values of h is of size $M \times M$, then this method requires $O(M^2N^2)$ time to compute.

Speedup at the risk of possible failure to find the best h can be achieved by using a hill-climbing technique. This technique begins with an initial estimate $\mathbf{h}_0$ of the disparity. To obtain the next guess from the current guess $\mathbf{h}_k$, one evaluates the difference function at all points in a small (say, 3x3) neighborhood of $\mathbf{h}_k$ and takes as the next guess $\mathbf{h}_{k+1}$ that point which minimizes the difference function. As with all hill-climbing techniques, this method suffers from the problem of false peaks: the local optimum that one attains may not be the global optimum. This technique operates in $O(M^2N)$ time on the average, for M and N as above.

Another technique, known as the sequential similarity detection algorithm (SSDA) [2], only estimates the error for each disparity vector h. In SSDA, the error function must be a cumulative one such as the L_1 or L_2 norm. One stops accumulating the error for the current h under investigation when it becomes apparent that the current h is not likely to give the best match. Criteria for stopping include a fixed threshold such that when the accumulated error exceeds this threshold one goes on to the next h, and a variable threshold which increases with the number of pixels in R whose contribution to the total error have been added. SSDA leaves unspecified the order in which the h's are examined.

Note that in SSDA if we adopt as our threshold the minimum error we have found among the h examined so far, we obtain an algorithm similar to alpha-beta pruning in min-max game trees [7]. Here we take advantage of the fact that in evaluating $\min_{\mathbf{h}} \sum_{\mathbf{x}} d(\mathbf{x},\mathbf{h})$, where $d(\mathbf{x},\mathbf{h})$ is the contribution of pixel x at disparity h to the total error, the $\sum_{\mathbf{x}}$ can only increase as we look at more x's (more pixels).

Some registration algorithms employ a coarse-fine search strategy. See [6] for an example. One of the techniques discussed above is used to find the best registration for the images at low resolution, and the low resolution match is then used to constrain the region of possible matches examined at higher resolution. The coarse-fine strategy is adopted implicitly by some image understanding systems which work with a "pyramid" of images of the same scene at various resolutions.

It should be noted that some of the techniques mentioned so far can be combined because they concern orthogonal aspects of the image registration problem. Hill climbing and exhaustive search concern only the order in which the algorithm searches for the best match, and SSDA specifies only the method used to calculate (an estimate of) the difference function. Thus for example, one could use the SSDA technique with either hill climbing or exhaustive search. In addition a coarse-fine strategy may be adopted.

The algorithm we present specifies the order in which to search the space of possible h's. In particular, our technique starts with an initial estimate of h, and it uses the spatial intensity gradient at each point of the image to modify the current estimate of h to obtain an h which yields a better match. This process is repeated in a kind of Newton-Raphson iteration. If the iteration converges, it will do so in $O(M^2 \log N)$ steps on the average. This registration technique can be combined with a coarse-fine strategy, since it requires an initial estimate of the approximate disparity h.

4. The registration algorithm

In this section we first derive an intuitive solution to the one-dimensional registration problem, and then we derive an alternative solution which we generalize to multiple dimensions. We then show how our technique generalizes to other kinds of registration. We also discuss implementation and performance of the algorithm.

4.1. One dimensional case

In the one-dimensional registration problem, we wish to find the horizontal disparity h between two curves $F(x)$ and $G(x) = F(x+h)$. This is illustrated in figure 2.

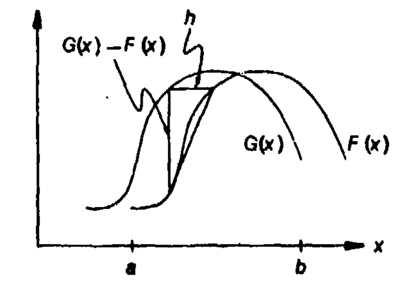


Figure 2: Two curves to be matched

Our solution to this problem depends on a linear approximation to the behavior of $F(x)$ in the neighborhood of x, as do all subsequent solutions in this paper. In particular, for small h,

$$F'(x) \approx \frac{F(x+h) - F(x)}{h} \tag{1}$$
$$= \frac{G(x) - F(x)}{h},$$

so that

$$h \approx \frac{G(x) - F(x)}{F'(x)}. \tag{2}$$

The success of our algorithm requires h to be small enough that this approximation is adequate. In section 4.3 we will show how to extend the range of h's over which this approximation is adequate by smoothing the images.

The approximation to h given in (2) depends on x. A natural method for combining the various estimates of h at various values of x would be to simply average them:

$$h \approx \sum_x \frac{G(x)-F(x)}{F'(x)} \Big/ \sum_x 1. \tag{3}$$

We can improve this average by realizing that the linear approximation in (1) is good where $F(x)$ is nearly linear, and conversely is worse where $|F''(x)|$ is large. Thus we could weight the contribution of each term to the average in (3) in inverse proportion to an estimate of $|F''(x)|$. One such estimate is

$$F''(x) \approx \frac{G'(x)-F'(x)}{h}. \tag{4}$$

Since our estimate is to be used as a weight in an average, we can drop the constant factor of 1/h in (4), and use as our weighting function

$$w(x) = \frac{1}{|G'(x)-F'(x)|}. \tag{5}$$

This in fact appeals to our intuition; for example, in figure 2, where the two curves cross, the estimate of h provided by (2) is 0, which is bad; fortunately, the weight given to this estimate in the average is small, since the difference between $F'(x)$ and $G'(x)$ at this point is large. The average with weighting is

$$h \approx \sum_x \frac{w(x)[G(x)-F(x)]}{F'(x)} \Big/ \sum_x w(x). \tag{6}$$

where $w(x)$ is given by (5).

Having obtained this estimate, we can then move $F(x)$ by our estimate of h, and repeat this procedure, yielding a type of Newton-Raphson iteration. Ideally, our sequence of estimates of h will converge to the best h. This iteration is expressed by

$$h_0 = 0,$$

$$h_{k+1} = h_k + \sum_x \frac{w(x)[G(x)-F(x+h_k)]}{F'(x+h_k)} \Big/ \sum_x w(x). \tag{7}$$

4.2. An alternative derivation

The derivation given above does not generalize well to two dimensions because the two-dimensional linear approximation occurs in a different form. Moreover, (2) is undefined where $F'(x) = 0$, i.e. where the curve is level. Both of these problems can be corrected by using the linear approximation of equation (1) in the form

$$F(x+h) \approx F(x)+hF'(x), \tag{8}$$

to find the h which minimizes the L_2 norm measure of the difference between the curves:

$$E = \sum_x [F(x+h)-G(x)]^2.$$

To minimize the error with respect to h, we set

$$0 = \frac{\partial E}{\partial h}$$

$$\approx \frac{\partial}{\partial h} \sum_x [F(x)+hF'(x)-G(x)]^2$$

$$= \sum_x 2F'(x)[F(x)+hF'(x)-G(x)],$$

from which

$$h \approx \frac{\sum_x F'(x)[G(x)-F(x)]}{\sum_x F'(x)^2}. \tag{9}$$

This is essentially the same solution that we derived in (6), but with the weighting function $w(x) = F'(x)^2$. As we will see, the form of the linear approximation we have used here generalizes to two or more dimensions. Moreover, we have avoided the problem of dividing by 0, since in (9) we will divide by 0 only if $F'(x) = 0$ *everywhere* (in which case h really is undefined), whereas in (3) we will divide by 0 if $F'(x) = 0$ *anywhere*.

The iterative form with weighting corresponding to (7) is

$$h_0 = 0,$$

$$h_{k+1} = h_k + \frac{\sum_x w(x)F'(x+h_k)[G(x)-F(x+h_k)]}{\sum_x w(x)F'(x+h_k)^2}, \tag{10}$$

where $w(x)$ is given by (5).

4.3. Performance

A natural question to ask is under what conditions and how fast the sequence of h_k's converges to the real h. Consider the case

$$F(x) = \sin x,$$

$$G(x) = F(x+h) = \sin(x+h).$$

It can be shown that both versions of the registration algorithm given above will converge to the correct h for $|h| < \pi$, that is, for initial misregistrations as large as one-half wavelength. This suggests that we can improve the range of convergence of the algorithm by suppressing high spatial frequencies in the image, which can be accomplished by smoothing the image, i.e. by replacing each pixel of the image by a weighted average of neighboring pixels. The tradeoff is that smoothing suppresses small details, and thus makes the match less accurate. If the smoothing window is much larger than the size of the object that we are trying to match, the object may be suppressed entirely, and so no match will be possible.

Since lowpass-filtered images can be sampled at lower resolution with no loss of information, the above observation suggests that we adopt a coarse-fine strategy. We can use a low-resolution smoothed version of the image to obtain an approximate match. Applying the algorithm to higher resolution images will refine the match obtained at lower resolution.

While the effect of smoothing is to extend the range of convergence, the weighting function serves to improve the accuracy of the approximation, and thus to speed up the convergence. Without weighting, i.e. with $w(x) = 1$, the calculated disparity h_1 of the first iteration of (10) with $F(x) = \sin x$ falls off to zero as the disparity approaches one-half wavelength. However, with $w(x)$ as in (5), the calculation of disparity is much more accurate, and only falls off to zero at a disparity very near one-half wavelength. Thus with $w(x)$ as in (5) convergence is faster for large disparities.

4.4. Implementation

Implementing (10) requires calculating the weighted sums of the quantities $F'G$, $F'F$, and $(F')^2$ over the region of interest R. We cannot calculate $F'(x)$ exactly, but for the purposes of this algorithm, we can estimate it by

$$F'(x) \approx \frac{F(x+\Delta x) - F(x)}{\Delta x},$$

and similarly for $G'(x)$, where we choose Δx appropriately small (e.g. one pixel). Some more sophisticated technique could be used for estimating the first derivatives, but in general such techniques are equivalent to first smoothing the function, which we have proposed doing for other reasons, and then taking the difference.

4.5. Generalization to multiple dimensions

The one-dimensional registration algorithm given above can be generalized to two or more dimensions. We wish to minimize the L_2 norm measure of error:

$$E = \textstyle\sum_{\mathbf{x}\in R} [F(\mathbf{x}+\mathbf{h}) - G(\mathbf{x})]^2,$$

where $\mathbf{x}$ and $\mathbf{h}$ are n-dimensional row vectors. We make a linear approximation analogous to that in (8),

$$F(\mathbf{x}+\mathbf{h}) \approx F(\mathbf{x}) + \mathbf{h}\frac{\partial}{\partial \mathbf{x}}F(\mathbf{x}),$$

where $\partial/\partial\mathbf{x}$ is the gradient operator with respect to $\mathbf{x}$, as a column vector:

$$\frac{\partial}{\partial \mathbf{x}} = [\frac{\partial}{\partial x_1} \frac{\partial}{\partial x_2} \cdots \frac{\partial}{\partial x_n}]^T.$$

Using this approximation, to minimize E, we set

$$\begin{aligned} 0 &= \frac{\partial}{\partial \mathbf{h}}E \\ &\approx \frac{\partial}{\partial \mathbf{h}} \textstyle\sum_{\mathbf{x}} [F(\mathbf{x}) + \mathbf{h}\frac{\partial F}{\partial \mathbf{x}} - G(\mathbf{x})]^2 \\ &= \textstyle\sum_{\mathbf{x}} 2\frac{\partial F}{\partial \mathbf{x}}[F(\mathbf{x}) + \mathbf{h}\frac{\partial F}{\partial \mathbf{x}} - G(\mathbf{x})], \end{aligned}$$

from which

$$\mathbf{h} \approx \Big[\textstyle\sum_{\mathbf{x}} (\frac{\partial F}{\partial \mathbf{x}})^T [G(\mathbf{x}) - F(\mathbf{x})]\Big] \Big[\textstyle\sum_{\mathbf{x}} (\frac{\partial F}{\partial \mathbf{x}})^T (\frac{\partial F}{\partial \mathbf{x}})\Big]^{-1},$$

which has much the same form as the one-dimensional version in (9).

The discussions above of iteration, weighting, smoothing, and the coarse-fine technique with respect to the one-dimensional case apply to the n-dimensional case as well. Calculating our estimate of $\mathbf{h}$ in the two-dimensional case requires accumulating the weighted sum of five products ($(G-F)F_x$, $(G-F)F_y$, F_x^2, F_y^2, and F_xF_y) over the region R, as opposed to accumulating one product for correlation or the L_2 norm. However, this is more than compensated for, especially in high-resolution images, by evaluating these sums at fewer values of $\mathbf{h}$.

4.6. Further generalizations

Our technique can be extended to registration between two images related not by a simple translation, but by an arbitrary linear transformation, such as rotation, scaling, and shearing. Such a relationship is expressed by

$$G(\mathbf{x}) = F(\mathbf{x}\mathbf{A} + \mathbf{h}),$$

where $\mathbf{A}$ is a matrix expressing the linear spatial transformation between $F(\mathbf{x})$ and $G(\mathbf{x})$. The quantity to be minimized in this case is

$$E = \textstyle\sum_{\mathbf{x}} [F(\mathbf{x}\mathbf{A} + \mathbf{h}) - G(\mathbf{x})]^2.$$

To determine the amount $\Delta\mathbf{A}$ to adjust $\mathbf{A}$ and the amount $\Delta\mathbf{h}$ to adjust $\mathbf{h}$, we use the linear approximation

$$\begin{aligned} &F(\mathbf{x}(\mathbf{A}+\Delta\mathbf{A}) + (\mathbf{h}+\Delta\mathbf{h})) \\ &\qquad \approx F(\mathbf{x}\mathbf{A}+\mathbf{h}) + (\mathbf{x}\Delta\mathbf{A} + \Delta\mathbf{h})\frac{\partial}{\partial \mathbf{x}}F(\mathbf{x}). \qquad (11) \end{aligned}$$

When we use this approximation the error expression again becomes quadratic in the quantities to be minimized with respect to. Differentiating with respect to these quantities and setting the results equal to zero yields a set of linear equations to be solved simultaneously.

This generalization is useful in applications such as stereo vision, where the two different views of the object will be different views, due to the difference of the viewpoints of the cameras or to differences in the processing of the two images. If we model this difference as a linear transformation, we have (ignoring the registration problem for the moment)

$$F(x) = \alpha G(x) + \beta,$$

where α may be thought of as a contrast adjustment and β as a brightness adjustment. Combining this with the general linear transformation registration problem, we obtain

$$E = \sum_x [F(xA + h) - (\alpha G(x) + \beta)]^2$$

as the quantity to minimize with respect to α, β, A, and h. The minimization of this quantity, using the linear approximation in equation (11), is straightforward. This is the general form promised in section 2. If we ignore A, minimizing this quantity is equivalent to maximizing the correlation coefficient (see, for example, [3]); if we ignore α and β as well, minimizing this form is equivalent to minimizing the L_2 norm.

5. Application to stereo vision

In this section we show how the generalized registration algorithm described above can be applied to extracting depth information from stereo images.

5.1. The stereo problem

The problem of extracting depth information from a stereo pair has in principle four components: finding objects in the pictures, matching the objects in the two views, determining the camera parameters, and determining the distances from the camera to the objects. Our approach is to combine object matching with solving for the camera parameters and the distances of the objects by using a form of the fast registration technique described above.

Techniques for locating objects include an interest operator [6], zero crossings in bandpass-filtered images [5], and linear features [1]. One might also use regions found by an image segmentation program as objects.

Stereo vision systems which work with features at the pixel level can use one of the registration techniques discussed above. Systems whose objects are higher-level features must use some difference measure and some search technique suited to the particular feature being used. Our registration algorithm provides a stereo vision system with a fast method of doing pixel-level matching.

Many stereo vision systems concern themselves only with calculating the distances to the matched objects. One must also be aware that in any real application of stereo vision the relative positions of the cameras will not be known with perfect accuracy. Gennery [4] has shown how to simultaneously solve for the camera parameters and the distances of objects.

5.2. A mathematical characterization

The notation we use is illustrated in figure 3. Let c be the vector of camera parameters that describe the orientation and position of camera 2 with respect to camera 1's coordinate system. These parameters are azimuth, elevation, pan, tilt, and roll, as defined in [4]. Let x denote the position of an image in the camera 1 film plane of an object. Suppose the object is at a distance z from camera 1. Given the position in picture 1 x and distance z of the object, we could directly calculate the position p(x,z) that it must have occupied in three-space. We express p with respect to camera 1's coordinate system so that p does not depend on the orientation of camera 1. The object would appear on camera 2's film plane at a position q(p,c) that is dependent on the object's position in three-space p and on the camera parameters c. Let $G(x)$ be the intensity value of pixel x in picture 1, and let $F(q)$ the intensity value of pixel q in picture 2. The goal of a stereo vision system is to invert the relationship described above and solve for c and z, given x, F and G.

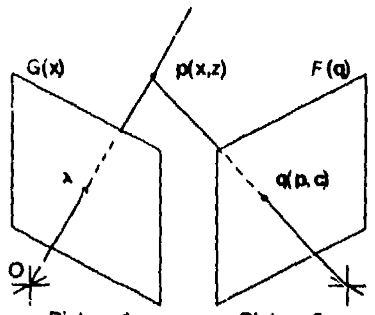


Figure 3: Stereo vision

5.3. Applying the registration algorithm

First consider the case where we know the exact camera parameters c, and we wish to discover the distance z of an object. Suppose we have an estimate of the distance z. We wish to see what happens to the quality of our match between F and G as we vary z by an amount Δz. The linear approximation that we use here is

$$F(z + \Delta z) \approx F(z) + \Delta z \frac{\partial F}{\partial z},$$

where

$$\frac{\partial F}{\partial z} = \frac{\partial p}{\partial z} \frac{\partial q}{\partial p} \frac{\partial F}{\partial q}. \tag{12}$$

This equation is due to the chain rule of the gradient operator; $\partial q/\partial p$ is a matrix of partial derivatives of the components of q with respect to the components of p, and $\partial F/\partial q$ is the spatial intensity gradient of the image $F(q)$. To update our estimate of z, we want to find the Δz which satisfies

$$0 = \frac{\partial}{\partial \Delta z} E$$
$$\approx \frac{\partial}{\partial \Delta z} \Sigma_x [F + \Delta z \frac{\partial F}{\partial z} - G]^2.$$

Solving for Δz, we obtain

$$\Delta z \approx \Sigma_x \frac{\partial F}{\partial z}[G - F] / \Sigma_x (\frac{\partial F}{\partial z})^2,$$

where $\partial F / \partial z$ is given by (12).

On the other hand, suppose we know the distances z_i, $i = 1,2,\ldots,n$, of each of n objects from camera 1, but we don't know the exact camera parameters $\mathbf{c}$. We wish to determine the effect of changing our estimate of the camera parameters by an amount $\Delta \mathbf{c}$. Using the linear approximation

$$F(\mathbf{c} + \Delta \mathbf{c}) \approx F(\mathbf{c}) + \Delta \mathbf{c} \frac{\partial \mathbf{q}}{\partial \mathbf{c}} \frac{\partial F}{\partial \mathbf{q}},$$

we solve the minimization of the error function with respect to $\Delta \mathbf{c}$ by setting

$$0 = \frac{\partial}{\partial \Delta \mathbf{c}} \Sigma_i \Sigma_{x \in R_i} [F(\mathbf{c} + \Delta \mathbf{c}) - G]^2$$
$$\approx \frac{\partial}{\partial \Delta \mathbf{c}} \Sigma_i \Sigma_x [F + \Delta \mathbf{c} \frac{\partial \mathbf{q}}{\partial \mathbf{c}} \frac{\partial F}{\partial \mathbf{q}} - G]^2,$$

obtaining

$$\Delta \mathbf{c} \approx [\Sigma_x (\frac{\partial \mathbf{q}}{\partial \mathbf{c}} \frac{\partial F}{\partial \mathbf{q}})^T [G - F]] [\Sigma_x (\frac{\partial \mathbf{q}}{\partial \mathbf{c}} \frac{\partial F}{\partial \mathbf{q}})^T (\frac{\partial \mathbf{q}}{\partial \mathbf{c}} \frac{\partial F}{\partial \mathbf{q}})]^{-1}.$$

As with the other techniques derived in this paper, weighting and iteration improve the solutions for Δz and $\Delta \mathbf{c}$ derived above.

5.4. An Implementation

We have implemented the technique described above in a system which functions well under human supervision. Our program is capable of solving for the distances to the objects, the five camera parameters described above, and a brightness and contrast parameter for the entire scene, or any subset of these parameters. As one would expect from the discussion in section 4.3, the algorithm will converge to the correct distances and camera parameters when the initial estimates of the z_i's and $\mathbf{c}$ are sufficiently accurate that we know the position in the camera 2 film plane of each object to within a distance on the order of the size of the object.

A session with this program is illustrated in figures 4 through 10. The original stereo pair is presented in figure 4. (Readers who can view stereo pairs cross-eyed will want to hold the pictures upside down so that each eye receives the correct view). The camera parameters were determined independently by hand-selecting matching points and solving for the parameters using the program described in [4].

Figures 5 and 6 are bandpass-filtered versions of the pictures in figure 4. Bandpass-filtered images are preferred to lowpass-filtered images in finding matches because very low spatial frequencies tend to be a result of shading differences and carry no (or misleading) depth information. The two regions enclosed in rectangles in the left view of figure 5 have been hand-selected and assigned an initial depth of 7.0 in units of the distance between cameras. If these were the actual depths, the corresponding objects would be found in the right view at the positions indicated figure 5. After seven depth-adjustment iterations, the program found the matches shown in figure 6. The distances are 6.05 for object 1 and 5.86 for object 2.

Figures 7 and 8 are bandpass-filtered with a band one octave higher than figures 5 and 6. Five new points have been hand-selected in the left view, reflecting the different features which have become visible in this spatial frequency range. Each has been assigned an initial depth equal to that found for the corresponding larger region in figure 6. The predicted position corresponding to these depths is shown in the right view of figure 7. After five depth-adjustment iterations, the matches shown in figure 8 were found. The corresponding depths are 5.96 for object 1, 5.98 for object 2, 5.77 for object 3, 5.78 for object 4, and 6.09 for object 5.

Figures 9 and 10 are bandpass-filtered with a band yet another octave higher than figures 7 and 8. Again five new points have been hand-selected in the left view, reflecting the different features which have become visible in this spatial frequency range. Each has been assigned an initial depth equal to that found for the corresponding region in figure 8. The predicted position corresponding to these depths is shown in the right view of figure 9. After four depth-adjustment iterations, the matches shown in figure 10 were found. The corresponding depths are 5.97 for object 1, 5.98 for object 2, 5.80 for object 3, 5.77 for object 4, and 5.98 for object 5.

5.5. Future research

The system that we have implemented at present requires considerable hand-guidance. The following are the issues we intend to investigate toward the goal of automating the process.

- Providing initial depth estimates for objects: one should be able to use approximate depths obtained from low-resolution images to provide initial depth estimates for nearby objects visible only at higher resolutions. This suggests a coarse-fine paradigm not just for the problem of finding individual matches but for the problem of extracting depth information as a whole.

- Constructing a depth map: one could construct a depth map from depth measurements by some interpolation method, and refine the depth map with depth measurements obtained from successively higher-resolution views.

- Selecting points of interest: the various techniques mentioned in section 3 should be explored.

- Tracking sudden depth changes: the sudden depth changes found at the edges of objects require some set of higher-level heuristics to keep the matching algorithm on track at object boundaries.

- Compensating for the different appearances of objects in the two views: the general form of the matching algorithm that allows for arbitrary linear transformations should be useful here.

Acknowledgements

We would like to thank Michael Horowitz, Richard Korf, and Pradeep Sindhu for their helpful comments on early drafts of this paper.

References

1. Baker, H. Harlyn. Edge Based Stereo Correlation. DARPA Image Understanding Workshop, April, 1980, pp. 168-175.

2. Barnea, Daniel I. and Silverman, Harvey F. "A Class of Algorithms for Fast Digital Image Regis ation." *IEEE Transactions on Computers C-21*, 2 (February 1972), 1/9-186.

3. Dudewicz, Edward J. *Introduction to Statistics and Probability.* Holt, Rinehart and Winston, New York, 1976.

4. Gennery, Donald B. Stereo-Camera Calibration. DARPA Image Understanding Workshop, November, 1979, pp. 101-107.

5. Marr, D. and Poggio, T. "A Computational Theory of Human Stereo Vision." *Proceedings of the Royal Society of London B-204* (1979), 301-328.

6. Moravec, Hans. P. Visual Mapping by a Robot Rover. Sixth International Joint Conference on Artificial Intelligence, Tokyo, August, 1979, pp. 598-600.

7. Nilsson, Nils J. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, New York, 1971.

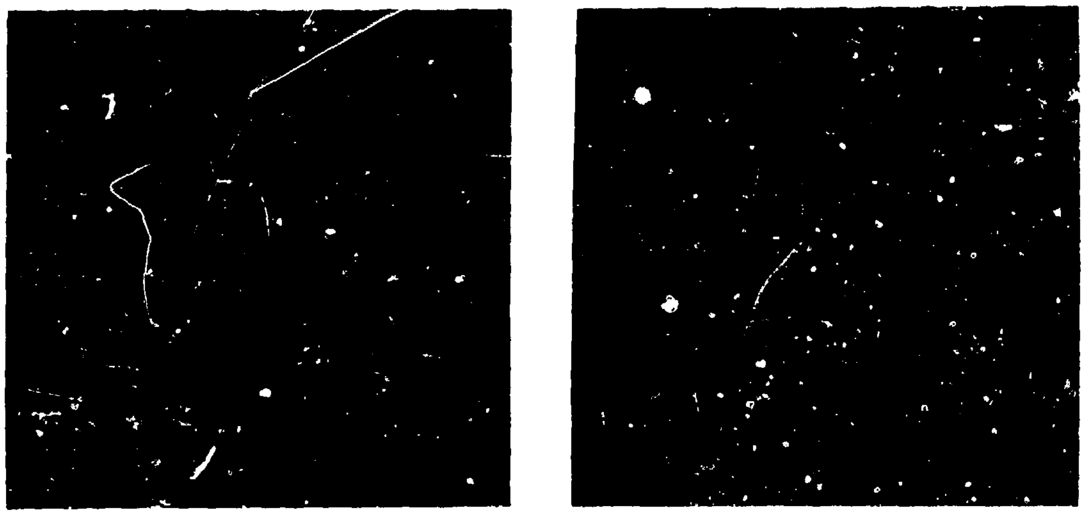


Figure 4.

Figure 5.

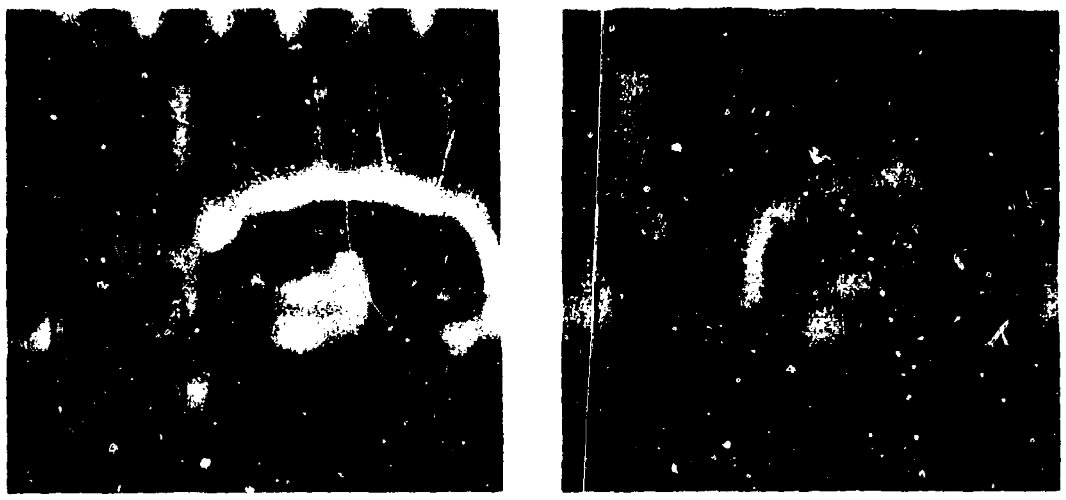


Figure 6.

Figure 7.

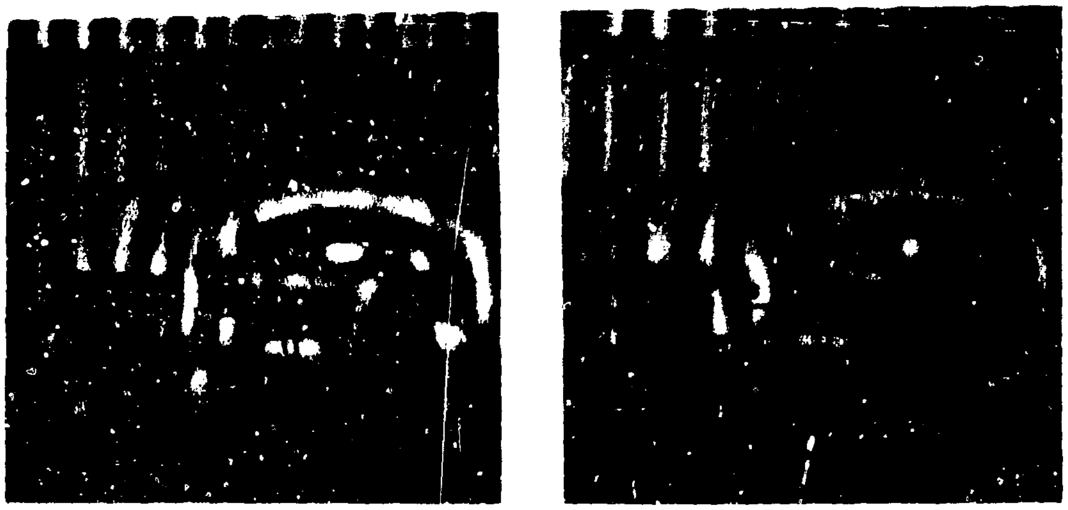


Figure 8.

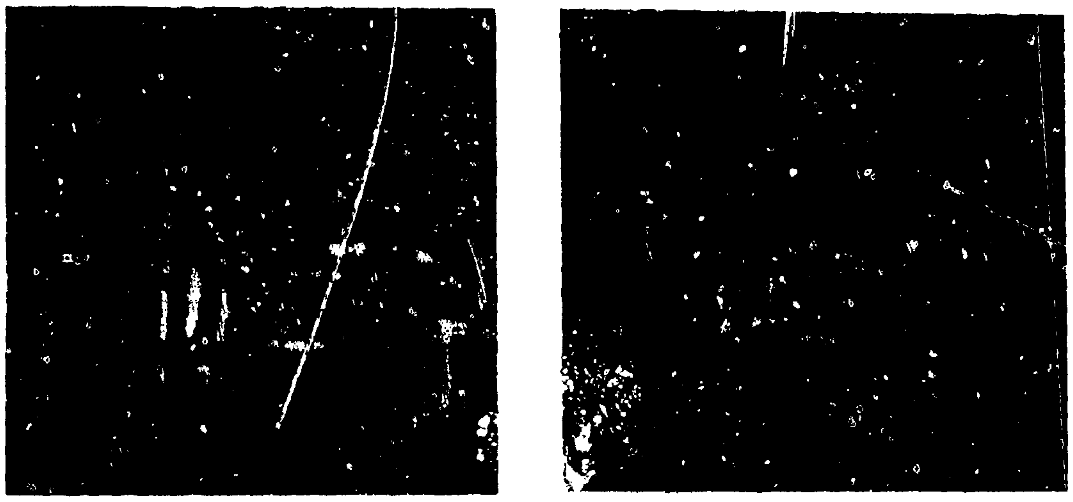


Figure 9.

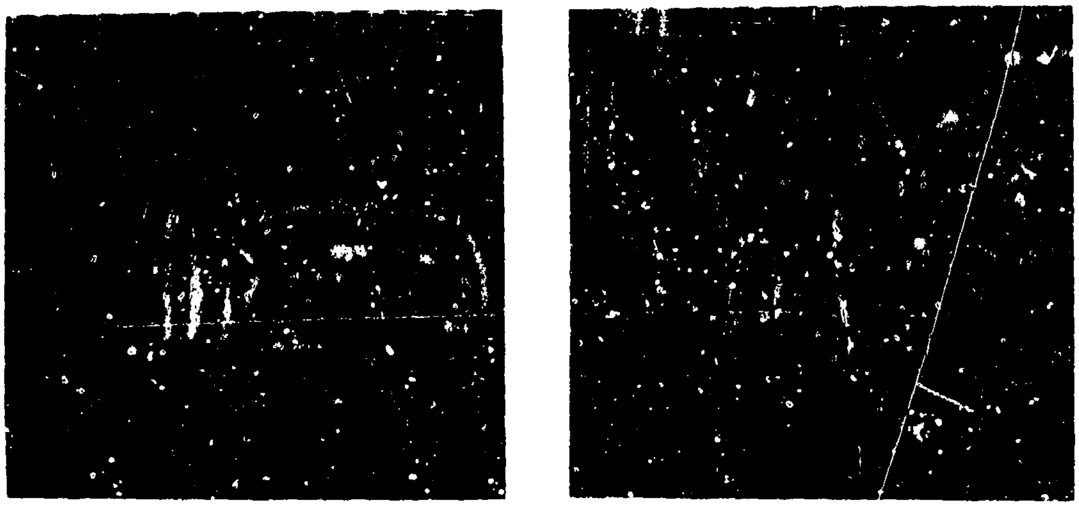


Figure 10.

BOOTSTRAP STEREO ERROR SIMULATIONS

Marsha Jo Hannah

Lockheed Palo Alto Research Laboratory
Department 52-53, Building 201
3251 Hanover Street
Palo Alto, CA 94304

ABSTRACT

Over the past three years, Lockheed has been working in navigation of an autonomous aerial vehicle using passively sensed images. One technique which has shown promise is bootstrap stereo, in which the vehicle's position is determined from the perceived locations of known ground control points. Successive pairs of known vehicle camera positions are then used to locate corresponding image points on the ground, creating new control points. This paper describes a series of error simulations which have been performed to investigate the error propagation as the number of bootstrapping iterations increases.

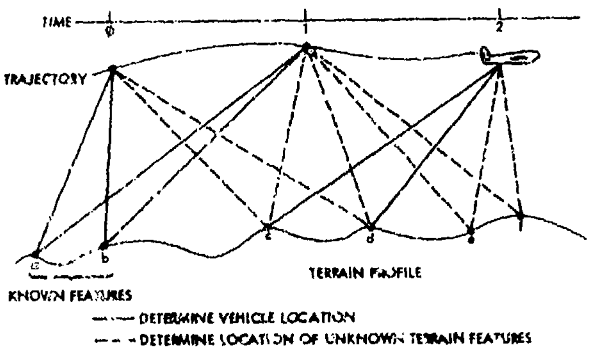
INTRODUCTION

A previous paper [1] postulated an autonomous navigation system, called the Navigation Expert, which attempts to approximate the sophistication of an early barnstorming pilot. This expert would navigate partly by its simple instruments (altimeter, airspeed indicator, and attitude gyros), but mostly by what it could see of the terrain below it. A major component of the Navigation Expert is a technique which we called bootstrap stereo.

Given a set of ground control points with known real-world positions, and given the locations of the projections of these points onto the image plane, it is possible to determine the position and orientation of the camera which collected the image. Conversely, given the positions and orientations of two cameras and the locations of corresponding point-pairs in the two image planes, the real-world locations of the viewed ground points can be determined [2]. Combining these two techniques iteratively produces the basis for bootstrap stereo.

Figure 1 shows an Autonomous Aerial Vehicle which has obtained images at three points in its trajectory. The bootstrap stereo process begins with a set of landmark points, simplified here to the two points a and b, whose real-world coordinates are known. From these, the camera position and orientation is determined for the image frame taken at Time 0. Standard image-matching correlation techniques [3] are then used to locate these same points in the second, overlapping frame taken at Time 1. This permits the second camera position and orientation to be determined.

Because the aircraft will soon be out of sight of the known landmarks, new landmark points must be established whenever possible. For this purpose, "interesting points"--points whose surrounding information indicates a high likelihood of their being matchable [4]--are selected in the first image and matched in the second image. Successfully matched points have their real-world locations calculated from the camera position and orientation data, then join the landmarks list. In Figure 1, landmarks c and d are located in this manner at Time 1; these new points are later used to position the aircraft at Time 2. Similarly, at Time 2, new landmarks e and f join the list; old landmarks a and b, which are no longer in the field-of-view, are dropped from the landmarks list.


TIME
0
1
2
TRAJECTORY
a
b
c
d
e
f
TERRAIN PROFILE
KNOWN FEATURES
DETERMINE VEHICLE LOCATION
DETERMINE LOCATION OF UNKNOWN TERRAIN FEATURES

TIME	POSITION OF CRAFT DERIVED FROM	DETERMINE POSITION OF
0	a, b	–
1	a, b	c, d
2	c, d	e, f

Figure 1 Navigation Using Bootstrap Stereo.

Bootstrap stereo, then, consists of four components--camera calibration, new landmark point selection, point matching, and control point positioning. All of these components are well established in the photogrammetry and image processing literature; each is conceded to work reasonably well. What was not known was how the errors would accumulate and propagate in an iterative application such as bootstrap stereo. This paper describes a series of error simulations which have been performed to investigate the error buildup as the number of bootstrapping iterations increases.

ERROR SIMULATIONS

Ideally, the error analysis of a new technique would be performed on several representative sequences of real data with known ground truth. For bootstrap stereo, this would require a sequence of 50 to 100 images, with each consecutive pair having an overlap of approximately 75%. In addition, a set of known ground control points visible (and recognizable) in the first image of the sequence is needed to initialize the technique. It is desirable to have the position and orientation of the camera known for each image; this simplifies determining the buildup of camera position error. Alternatively, having sets of known, visible, recognizable ground control points in every third or fourth image would permit calculation of ground point position errors, which should be equivalent to camera position errors. Finally, it is imperative that the spatial distortion which the camera induces in the image be known, either analytically or empirically, so that the image pixel positions can be corrected for this distortion.

In the course of developing bootstrap stereo, it became obvious that we did not have a data set which met these requirements, nor could we obtain one within our available time and financial resources. Despite this, we needed to document the buildup of error in the camera and ground-point positions as the bootstrapping progressed. The only solution was to program a means for simulating a flight, thus creating data on which the pieces of code could operate as they would for actual bootstrapping.

The ideal manner in which to do this would be to simulate grey-level imagery of a realistic 3-dimensional surface, as seen from arbitrary viewpoints. This could be accomplished by creating a digital model of a region of terrain--complete with features such as vegetation, roads, and houses, as well as the reflectance properties of each part of the model. We could then draw up a flight path over the simulated terrain and calculate the set of images that the simulated camera would take of its simulated world as it moved along this path. These images could then be fed directly into the interesting-point program, etc., and we could easily compare the resulting bootstrapped positions to the simulated flight path. This approach, however, was deemed to be impractical to implement within our available computational and manpower resources.

We therefore re-examined what it was that we really needed to simulate. The major sources of error in bootstrapping come from errors in point matching between images and the manner in which these errors perturb the numerical analysis and projective geometry of the camera model calculations and the ground point positioning. If we could reasonably simulate the matched image points, complete with errors, we would not need simulated grey-level imagery.

Given this simplification, we proceeded with our simulation. We first created a digital terrain model by constructing a planar grid below the general swath of the flight path, then used a random number generator to create elevations at each point of the grid. We devised a flight path by flying our hypotehtical aircraft along a given vector, taking its position at intervals, and introducing random perturbations in the postion and orientation of the aircraft (hence of the camera) at each step along the way.

For each camera position, we performed the necessary projective geometry to determine where each of the terrain grid points fell in the image; those which fell outside of the field-of-view of the camera were discarded. Each image point was perturbed by a random amount and/or rounded (e.g. to the nearest pixel or 1/10 pixel) to simulate a match error. Image points were tagged with the ID numbers of the grid points which generated them, so that points in two images could be matched symbolically. (Figure 2 summarizes the parameters which define a simulation).

We then proceeded to run the bootstrapping programs on this data set. The programs for locating interesting points and matching them from image to image were replaced with a single program to retrieve interesting (i.e. visible) points from the image point files and match them from file to file by their ID numbers. The camera position calculation program and the ground-point positioning program were used without changes.

RESULTS

About 25 different experiments were done using the simplified simulation described above. These investigated the effects of various mission parameters (such as camera field-of-view, pointing angle with respect to the direction of flight, etc.) as well as the effects of match accuracy on the resulting errors in the camera position estimates.

The general conclusions from these simulations are that the distance traveled before path estimation error becomes unacceptable can be increased by:

1) Increasing the camera field-of-view (Fig. 3)
2) Increasing the platform stability (Fig. 4)
3) Increasing the match accuracy (Fig. 5)
4) Using backward-looking imagery (Fig. 6)

The first three of these are fairly obvious. Increasing the field-of-view increases the distance which can be traveled between camera shutterings while still maintaining 75% overlap in the data. Increasing the platform stability makes it less likely that wild swings in the pointing angle of the camera will decrease the ground coverage overlap, which increases camera positioner inaccuracy. Increasing the match accuracy decreases the uncertainty about camera and ground-point positions, allowing error to build up more slowly.

That backward-looking imagery should be superior to nadir or forward-looking imagery is not intuitively obvious. To understand this, consider the two major ways in which errors enter into camera positioning. If the set of ground data points and their corresponding image points are slightly inconsistent, the camera calibrator will make an error in the camera position and orientation. On the other hand, if a pair of camera positions and of matching points within the images are slightly inconsistent, then the rays from the camera center through the image points will not intersect precisely, giving an error in that ground-point position. Of these two errors, the ground-point positioning is more sensitive, since it depends on only two rays, while the camera position determination uses information from a larger number of ground-point-to-camera rays; the redundancy in the multiple observations greatly helps in reducing the error.

Now consider the geometry involved in the forward- and backward-looking cases. When the camera is looking forward, the new points are being placed far ahead of the camera by means of a very oblique triangle (Figure 7a), where a small error in match or camera orientation can cause a large error in the ground-point position. When the camera is looking backward, the new points are being placed almost directly under the camera, by means of a nearly equilateral triangle (Figure 7b)--the most favorable geometry for minimizing ground-point position error. Nadir imagery shares this favorable geometry, but suffers because of the small amount of visible terrain. Tipping the camera forward or backward brings more terrain into the field-of-view, permitting longer moves between images. Thus, of the three look orientations, backward-looking stereo provides the best combination of conditions to maximize the distance moved before the errors become unacceptable.

The obvious tactical question is how far can the bootstrap technique fly before its errors become unacceptable. That, of course, will depend on the flight parameters. We ran one simulation in which all parameters were favorably set; after flying 100,000 ft (almost 20 miles), the position was off by about 25 feet, and the error was still accumulating slowly. We do not know how far this flight could have gone before the errors became unacceptable, as this is the longest flight we have simulated.

It should be mentioned that most of our simulations did NOT include any use of the instrumentation on our simulated aircraft. Of course, any reasonable system which is flown will have attitude and altitude instruments whose readings at the times the camera is shuttered will be available to the processing system. These readings can provide good initial values for the solution of the highly nonlinear camera equations and can prevent bad data points from unduly perturbing the solution. We have flown one simulation using postulated instrument readings and constraining the camera position and orientation solution to lie near these. This simulation showed a 5-fold increase in distance traveled before the position became inaccurate, when compared with a similar run without the instrumentation and constraints.

CONCLUSIONS

When an autonomous aerial vehicle must navigate without using external signals or radiating energy, a visual navigator is an enticing possibility. We have proposed a Navigation Expert capable of emulating the behavior of an early barnstorming pilot in using terrain imagery. One tool such a Navigation Expert could use is bootstrap stereo. This is a technique by which the vehicle's position is determined from the perceived positions of known landmarks, then two known camera positions are used to locate real-world points which serve as new landmarks.

The components of bootstrap stereo are well established in the photogrammetry and image processing literature. We have combined these, with improvements, into a workable system [1]. Simulation results on the error buildup in the system have been encouraging.

Of course, these are still only simulation results. Until we can obtain calibrated, controlled imagery with known ground truth on which to run bootstrapping, then compare its results with a simulation having the same parameters, it will be difficult to tell how accurately our simulations represent the bootstrapping process.

REFERENCES

[1] Hannah, M. J., "Bootstrap Stereo", Proceedings: Image Understanding Workshop, College Park, Maryland, April, 30, 1980.

[2] Thompson, M. M., Manual of Photogrammetry, American Society of Photogrammetry, Falls Church, Virginia, 1944.

[3] Hannah, M. J., Computer Matching of Areas in Stereo Imagery, Ph.D. Thesis, AIM# 239, Computer Science Department, Stanford University, California, 1974.

[4] Moravec, H. P., "Visual Mapping by a Robot Rover", Proceedings of the 6th IJCAI, Tokyo, Japan, 1979.

	CHARACTERISTICS	PARAMETERS
TERRAIN:	x, y Grid points, with random elevations above a level plane	• Ground grid spacing • Base plane elevation • Amplitude of elevation
FLIGHT PATH:	Straight, level course, with random perturbations in position	• Vehicle elevation • Amplitude of position perturbations
CAMERA ORIENTATION:	Fixed with respect to the vehicle, with random perturbations in vehicle attitude	• Pitch angle of camera with respect to flight path • Amplitude of attitude perturbations (heading, pitch, and roll)
SYNTHETIC IMAGERY:	Image plane projections of terrain points, with random perturbations in image plane position	• Image plane size (x,y) • Field-of-view angle • Imagery overlap • Amplitude of image plane perturbations
PROCESSING:	Normal camera modeling and ground point positioning, with symbolic image point matching	• Number of iterations performed

Fig. 2 Characteristics and Parameters of the Bootstrap Stereo Simulations

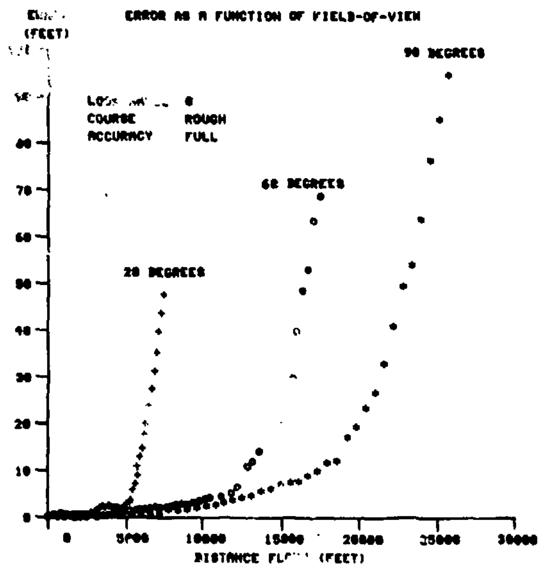


Figure 3 Error as a Function of Field-of-View.

Increasing the camera field-of-view increases the distance which can be flown before errors become unacceptable. (In this and subsequent simulations presented here, elevation = 1200 ft, distance flown was varied to maintain 75% overlap).

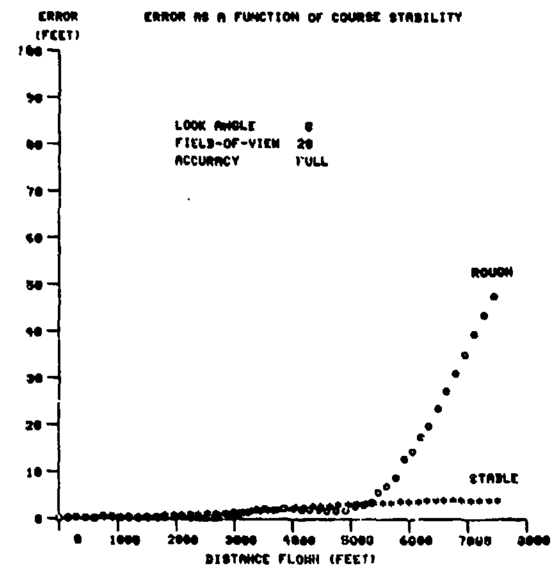


Figure 4 Error as a Function of Course Stability.

Increasing the platform stability decreases the error buildup, especially for narrow field-of-view cameras.

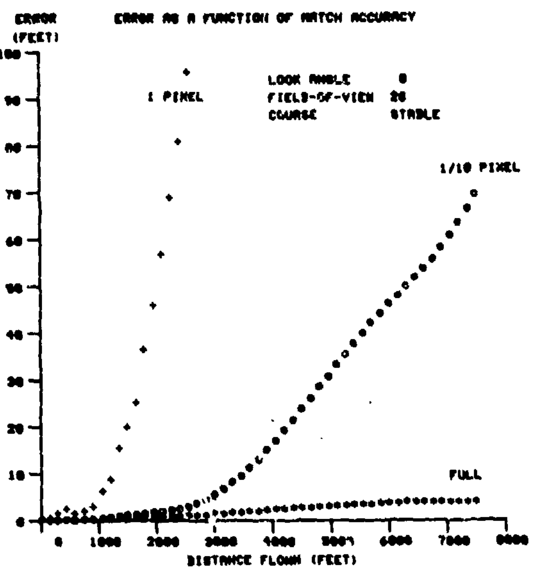


Figure 5 Error as a Function of Match Accuracy.

Increasing the match accuracy decreases the error buildup.

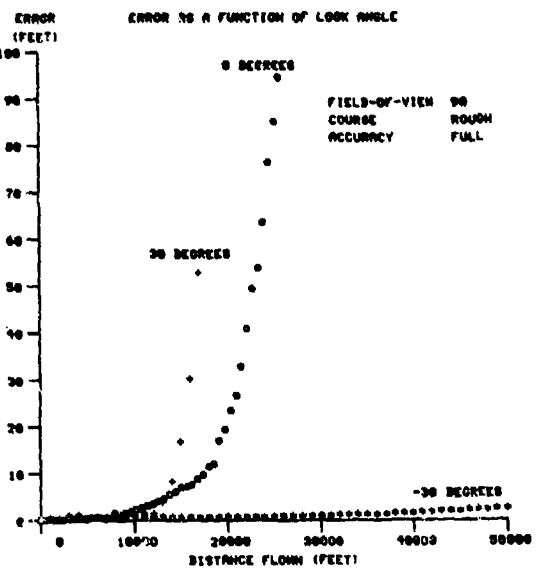


Figure 6 Error as a Function of Look Angle.

Using backward-looking imagery greatly increases the distance that can be flown before errors become unacceptable.

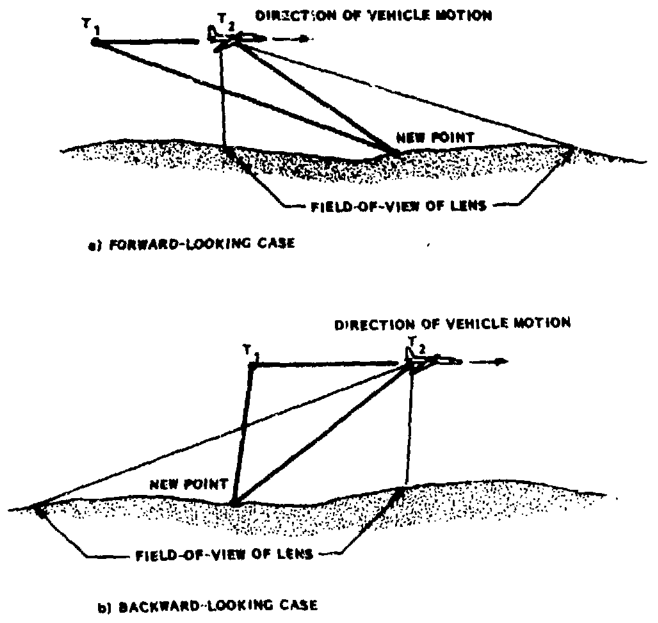


Figure 7 Effect of Forward- and Backward-Looking Camera.

The long, narrow triangle obtained in the forward-looking case is more sensitive to angular errors.

MODEL-BASED THREE DIMENSIONAL INTERPRETATIONS OF TWO DIMENSIONAL IMAGES

Rodney A. Brooks

Artificial Intelligence Laboratory
Stanford University, Stanford, Ca. 94305 USA

Abstract.

ACRONYM is a comprehensive domain independent model-based system for vision and manipulation related tasks. Many of its sub-modules and representations have been described elsewhere. Here the derivation and use of invariants for image feature prediction is described. We describe how predictions of image features and their relations are made and how instructions are generated which tell the interpretation algorithms how to make use of image feature measurments to derive three dimensional sizes and structural and spatial constraints on the original three-dimensional models. Some preliminary examples of ACRONYM's interpretations of aerial images are shown.

1. Introduction.

At the ARPA IU workshop of May 1980 we reported on a number of conceptual advances proposed as additions to the ACRONYM system. These have now all been implemented, and further extended in a number of cases. The major perfomance advantages are that now ACRONYM can discriminate instances of various classes of modesl, and extract three dimensional information from monocular images.

To support these devlopments we have added a class and subclass relation representation scheme to the geometric modeling system ([7], [5]). This is based on the use of symbolic algebraic constraints. In support of this a constraint manipulation systems which includes a partial decision procedure on consistency of sets of non-linear inequalities was formulated and implemented [5]. A geometric reasoning system which can deal with underconstrained spatial relations was developed [6]. A new matcher which could manipulate the constraint systems was built for interpretation [7]. All of these systems were implemented in a mixture of MACLISP and a new rule system built for the purpose.

We have thus moved from a purely geometric representation and qualitative geometric reasoning system to a system with a combined algebraic and geometric representation and a geometric reasoning system which can make precise deductions about partially specified situations. The geometric and algebraic aspects of the representation complement each other during interpretation.

In this paper we deal with the techniques developed for image feature and feature-relation prediction, and then give some first examples (February 1981) of the performance of the new incarnation of ACRONYM on some images. The low level processes we currently use provide either little or noisy data. Nevertheless ACRONYM makes strong and accurate deductions about the obejcts appearing in the images. We expect even better performance when more accurate low level descriptive processes become available.

2. Prediction.

In the ACRONYM system generic object classes and specific objects are represented by volumetric models based on generalised cones along with a partial order on sets of non-linear algebraic inequalities relating model parameters. Image features and relations between them which are invariant over variations in the models and camera parameters are identified by a geometric reasoning system. Such predictions are combined first to give guidance to low level image description processes, then to provide coarse filters on image features which are to be matched to local predictions. Predictions also contain instructions on how to use noisy measurements from identified image features to construct algebraic constraints on the original three dimensional models. Local matches are combined subject both to consistently meeting predicted image feature relations, and to the formation of consistent sets of algebraic constraints derived from the image. The result is a three dimensional interpretation of the image.

This section describes some of the invariants that are identified by the reasoning system, and gives examples of how the back constraints are set up giving three dimensional information about the instances of the models which appear in images.

2.1 Constraints.

To illuminate the discussion in succeeding subsections we briefly describe the uses and capabilities of ACRONYM's constraint mechanism and the allowed structure of constraints themselves.

ACRONYM's three-dimensional models are represented by *units* and *slots* (e.g. Bobrow and Winograd [3]). Any slot which admits numeric *fillers* also admits *quantifiers* (predeclared variable names) and expressions over quantifers using the operators $+$, $-$, $\times$, / and $\sqrt{\ }$.

Constraints can be put on quantifiers. They take the form of inequalities between expressions as defined above, along with the possibility of including max and min (on the left and right of $\leq$, respectively). Equality can be encoded as two inequalities. For instance suppose a cylinder is represented as a generalised cone whose straight spine has its length defined by the quantifier CYL_LENGTH and whose cross section is a circle with radius CYL_RADIUS. Then the class of all cylinders of volume 5 (in some units) can be represented by the two constraints:

$$5 \geq \mathtt{CYL\_LENGTH} \times \mathtt{CYL\_RADIUS} \times \mathtt{CYL\_RADIUS} \times \pi$$
$$5 \leq \mathtt{CYL\_LENGTH} \times \mathtt{CYL\_RADIUS} \times \mathtt{CYL\_RADIUS} \times \pi$$

The ACRONYM constraint manipulation system (CMS), described in detail in [5], operates on sets of constraints. A set of constraints (implicitly conjunctive) defines a subset of n-dimensional space, where n is the number of quantifiers mentioned in the constraint set, which is the set of points for which all constraints are true. This is called the satisfying set, and is empty if the constraints are inconsistent. The CMS is used for three tasks related to this constraint set.

1. Given a set of constraints partially decide whether their satisfying set is empty. The outcomes are "*empty*" or "*I don't know*".

2. Find numeric (or $\pm\infty$) upper and lower bounds on an expression in quantifers over the satisfying set of a constraint set. This uses procedures called SUP and INF.

3. (A generalization of 2.) For and expression E and a set of quantifiers V find expressions L and H in V such that $L \leq E \leq H$ identically over the satisfying set of the constraint set.

In 2 and 3 the expressions being bounded can include trigonometric functions such as sin, cos and arcsin. The CMS we have implemented in ACRONYM is a non-linear generalization of the linear SUP-INF method described by Bledsoe [[illegible]], and Shostak [9]. It behaves identically to that described by the latter for purely linear sets of constraints and linear expressions. In addition it can often produce good bounds (numeric and expressions) on highly non-linear expressions in the presence of many non-linear constraints.

2.2 Shape prediction

We predict shapes as *ribbons* (the two dimensional analogue of three dimensional generalized cones) and *ellipses*. These are also the features which are found by the low level descriptive process we are temporarily using in ACRONYM.

Ribbons are a good way of describing the images generated by generalised cones. Consider a ribbon which corresponds to the image of the swept surface of a generalized cone. For straight spines, the projection of the cone spine into the image would closely correspond to the spine of the ribbon. Thus a good approximation to the observed angle between the spines of two generalized cones is the angle between the spines of the two ribbons in the image corresponding to their swept surfaces. We do not have a quantitative theory of these correspondences. Ellipses are a good way of describing the shapes generated by the ends of generalized cones. The perspective projections of ends of cones with circular cross-sections are exactly ellipses.

Shape prediction involves deciding what shapes will be visible, predicting ranges for shape parameters (to be used as a coarse filter during interpretation and also to guide the low level descriptive processes) and deriving instructions about how to locally invert the perspective transform and hence use image measurements to generate constraints on the original three dimensional models.

To predict the shapes generated by a single generalized cone, we do not explicitly predict all possible qualitatively different viewpoints. Rather we predict what shapes may appear in the image, and associate with them methods to compute constraints on the model that are implied by their individual appearance in an the image. For example identification of the image of the swept surface of a right circular cone constrains the relative orientation of the cylinder to the camera (we call these back constraints). Identification of an end face of the cylinder provides a different set of constraints. If both the swept surface and an end face are identified then both sets of constraints apply. We also predict specific relations between shapes that will be true if they are both observed correctly. For more complex cones, the payoff is even greater for predicting individual shapes rather than exhaustive analysis of which shapes can appear together.

At other times during prediction invariant cases of obscuration are noticed. For instance it may be noticed that one cone abuts another so that its end face will never be visible. The consequences of such realizations are propagated through the predictions.

Prediction of shapes proceeds in five phases. First, all the contours on a generalised cone which could give rise to image shapes are indentified by a set of special purpose rules. These include occluding contours and contours due purely to internal cone faces. Thus for instance a right square cylinder will generate contours for the end faces, the swept faces, and contours generated by the swept edges at diagonally vertices of the square cross section. The contours are generated independently of camera orientation, and in terms of object dimensions rather than image quantities.

The orientation of the generalised cone relative to the camera (this is done by the geometric reasoning system, see [6], [5]) is then examined to decide which contours will be visible and how their image shapes will be distorted over the range of variations in the model parameters which appear in the orientation expressions.

The third phase predicts relations between contours of a single generalized cone (see section 2.3).

The actual shapes are then predicted. The expected values for shape parameters in the image are estimated as closed intervals (see below). Finally the back constraints which will be instantiated during interpretation are constructed.

2.2.1 Back constraints. Suppose that we wish to predict the length of an image feature which is generated by something of length l lying in a plane parallel to the camera image plane, at distance d from the camera. Furthermore suppose the camera has a focal ratio of f. Then the length of the image feature is given by $p = (l \times f)/d$. Any or all of l, f and d may be expressions in quantifiers, rather than numbers. Using the CMS we can obtain bounds on the above expression for image feature length, giving that it will lie in some range $P = [p_l, p_h]$ where p_l and p_h are either numbers or $\pm\infty$. For more complex geometries the expression for p will be more complex, but the method is the same (trigonometric functions are usually involved).

Now given an image feature, which is hypothesized to correspond to the prediction we have to decide whether it acceptable on the basis of its parameters. The low level descriptive processes are noisy and provide an error interval, rather than an exact measurement for image parameters. Suppose the interval is $M=[m_l, m_h]$ for a feature parameter predicted with expression p. Then the parameter is acceptable if $P \bigcap M$ is non-empty. This is the coarse filtering used during initial hypothesis of image feature to feature prediction matches.

But note also that it must be true that the true value of p for the particular instance of the model which is being imaged must lie in the range M. Thus we can add the constraints:

$$\begin{aligned} m_l &\leq (l \times f)/d \\ m_h &\geq (l \times f)/d \end{aligned}$$

to the instance of the model being hypothesized, where l, f and d are numbers or expressions in quantifiers.

2.2.2 Trigonometric back constraints. When the expression p involves trigonometric functions the above method of generating back constraints will not work. It would generate constraints involving trogonmetric functions, which our CMS can not handle.

One approach to this problem is to bound expression p above and below by expressions involving no quantifiers contained in arguments to trigonometric functions, and then use these expressions in setting up the back constraints. This has the unfortunate side effect of losing all information implied by the image feature about the quantifiers eliminated from the bounds.

A second approach is sometimes applicable. If a trigonometric function has as its argument e, an expression, and if the CMS determines that e is bounded to lie within a region of the function's domain where it is strictly monotonic and hence invertible, then specific back constraints on e can be computed at interpretation time (as distinct from during prediction). We illustrate with an example. A cylinder with length CYL_LENGTH is sitting upright on a table. A camera with unknown but constrained pan and tilt (the latter is constrained to lie in the interval $[\pi/12, \pi/6]$) is looking across from the side of the table, and it is elevated above table top height. The geometric details and numeric constants are not important here. Suffice it to say that the geometric reasoning system deduces that the pan of the camera is irrelevant to the prediction of the length of the ribbon corresponding to the swept surface of the cylinder. It predicts that the length of the ribbon in the image will in fact be:

$$\frac{-2.42 \times \mathtt{CYL\_LENGTH} \times \cos(-\mathtt{TILT})}{\mathtt{CYLINDER.CAMZ}}$$

where 2.42 is the focal ratio of the camera and CYLINDER.CAMZ is an internal quantifier generated by the prediction module.

Both of the above approaches are used to generate back constraints to ensure coverage of all the relevant quantifiers. They are:

$$\begin{aligned} m_h &\geq -2.096 \times \mathtt{CYL\_LENGTH} \times (1/\mathtt{CYLINDER.CAMZ}) \\ m_l &\leq -2.338 \times \mathtt{CYL\_LENGTH} \times (1/\mathtt{CYLINDER.CAMZ}) \\ -\mathtt{TILT} &\leq -\arccos(\sup(-0.413 \times m_h \\ &\qquad \times \mathtt{CYLINDER.CAMZ} \times (1/\mathtt{CYL\_LENGTH}))) \\ -\mathtt{TILT} &\geq -\arccos(\inf(-0.413 \times m_l \\ &\qquad \times \mathtt{CYLINDER.CAMZ} \times (1/\mathtt{CYL\_LENGTH}))) \end{aligned}$$

The first two are non-trigonometric back constraints and at interpertation time a simple susbsitution of the measured numeric quantities for m_l and m_h is done. The latter two require further comuptation at interpretation time. After the substitution, expressions must be bounded over the satisfying set of all the known constraints, and the function arccos applied to give numeric upper and lower bounds on the quantifier TILT.

The techniques described here work for a more general class of functions than trigonometric functions (in the current implementation of ACRONYM we use it for functions sin, cos and arcsin). The requirement is that the domain of the function (e.g. the interval $[-\pi, \pi]$ for sin and cos), can be subdivided into a finite number of intervals over which the function is strictly monotonic, and hence locally invertible.

2.3 Feature relation prediction.

Image feature (shape) predictions are organized as the nodes of the *prediction graph*. The arcs of the graph predict image-domain relations between the features. During interpretation pairs of hypothesized image feature to predicton node matches are coarsely checked for consistency by attempting to instantiate the prediction arcs. Some arcs also include back constraints which the instantiation of the arc implies about the model. These are treated in exactly the same manner as those associated with image feature predictions.

Prediction arcs are generated to relate multiple shapes predicted for a single cone. For instance a right circular cylinder prediction includes shapes for the swept surface and perhaps each of the end faces (depending on whether the camera geometry is known well enough to determine *a priori* exactly which faces will be visible). It can be predicted that a visble end face will be co-incident at at least one point in the image with a visible swept surface. (In fact a stronger prediction can be made: the straight spine of the swept surface image ribbon can be extended through the center of mass of the elliptical image of the end face.)

Prediction arcs are also generated between shapes associated with predictions for different generalised cones. These are actually of more importance in arriving at a consistent global interpretation of collections of image features as complex objects.

The semantics of the arc types we currently use are as follows.

<u>2.3.1 Exclusive</u>. If a generalised cone has a straight spine and during sweeping the cross section is kept at a constant angle to the spine the at most one of the cones end faces can be visible in a single image. *Exclusive* arcs relate image features which are mutually exclusive for this or other reasons. (Note that in the case instantiations of the two end faces would probably result in inconsistent back constraints being applied to the the spatial orientation of the original model, so that eventually the CMS would detect an inconsistency. However checking for the existence of a simple arc at an early stage is computationally much cheaper th[illegible] waiting to invoke the decision procedure.)

<u>2.3.2 Co-linear</u>. If two line segments in three-space are *co-linear* then any two-space image of them will either be a single degenerate point or two co-linear line segments. As we pointed out in the previous section a the spine of the image shape corresponding to the swept surface of a cone is usually a good approximation to the projection of the spine of the cone into the image. Thus if two cones are known to invariantly have co-linear spines in three dimensions a co-linear spine arc between the predictions of their swept surfaces can be included.

<u>2.3.3 Co-incident</u>. If two cones are physically *co-incident* at some point(s) in three-space then for any camera geometry, if they are both visible their projections will be co-incident at some point(s) (except for some cases of obscuration). Failure to match predicted co-incident arcs turns out to be the strongest pruning process during image interpretation.

<u>2.3.4 Angle</u>. If the *angle* between the spines of two generalized cones as viewed from the modeled camera is invariant over all the rotational variations in the model (e.g. wing-wing and wing-fuselage angles when an aircraft is viewed from above — this is because the only rotational freedom of an aircraft on the ground is about an axis parallel to the direction of view of an overhead camera), or if an expression for the observed angle can be symbolically computed and is suficiently simple, then a prediction of the observed angle can be made. Again the fact the projections of model spine correspond to image spines is used here. This arc type includes (trigonometric) back constraints which make use of the observed angle. Some such constraints constrain relative spatial orientations of generalised cones. Others provide constraints on the orientation of the plane of rotation, which generated the angle, relative to the camera, and hence constraints on an object's orientation relative to the camera.

<u>2.3.5 Projection</u>. Suppose one cone *B* is affixed at one end of its spine to another *A* somewhere along its length. The spines need not be co-incident, but the cones must be. Then the normal *projection* (we are not talking about projection in the imaging sense here) of the spine-end of *B* onto the spine of *A* defines a ratio between distance along the spine of *A* and the spine length of *A*. If the spines of *A* and *B* are both observable then the same projection in the image is invariant over all possible camera orientations. For example the ratio of the distance from the rear of the fueslage to the point of wing attachment, to the length of the fuselage, is invariant over all viewing angles. Again we rely on the coresspondences between the projection (other sense here) of a cone spine and the spine of the ribbon generated by the image of its swept surface. Projection arcs are only generated for pairs of image features which have a co-incident arc. They provide back constraints on the model via the symbolic expression which describes the modeled spine projection ratio.

<u>2.3.6 Distance</u>. Sometimes symbolic expressions for the image distance between two image features can computed. *Distance* arcs are only generated for pairs of image features which also have an angle arc, but no co-incident arc. Distance arcs generate back constraints on the original model.

<u>2.3.7 Ribbon-contains</u>. This is a directed arc type which relates two predicted ribbons, one of which will two dimensionally contain the other in the image. For instance, *ribbon-contains* arcs are built between the ribbon predicted from the occluding contour of a generalized cone with rectangular cross-section, and each of the ribbons generated by the two visible swept faces.

3. Some Image Interpretations.

At the time of writing the various sub-systems have only been running together for about two weeks. The image interpretations reported here are therefore of a rather preliminary nature.

In the examples to be described here ACRONYM was given a generic model of wide-bodied passenger jet aircraft, along with class specialisations to L-1011s and Boeing-747s. The Boeing-747 class had further subclass specialisations to Boeing-747B and Boeing-747SP. The subclasses are do not completely partition their parent classes. The classes are described by sets of constraints on some 30 quantifiers. Figure 3.1 shows instances of the two major modeled classes of jet aircraft. These diagrams were draw by ACRONYM

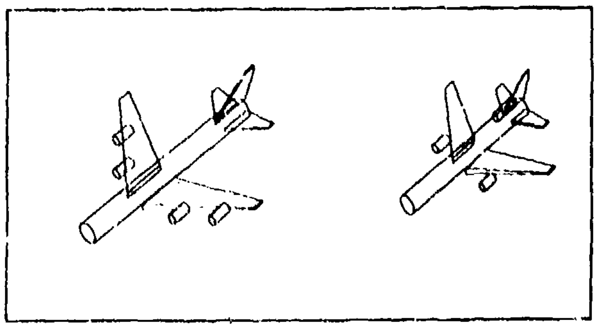


Fig. 3.1: Instances of class models of Boeing-747s and L-1011s.

from the models given it to carry out the image interpretations. The constraints for the generic class of wide bodied jets are given in figure 3.2. Units are meters.

The camera was modeled as being between 1000 and 12000 meters above the ground. Thus there is little *a priori* knowledge of the scale of the images. A specific focal ratio was given: 20. (Similar interpretations have been carried out with a variable focal ratio, but then the final constraints on camera height and focal ratio are coupled, and not as clear for illustrative purposes — no accuracy is lost due to the non-linearities that are introduced into the constraints, although both computation time and garbage collection time are increased.)

The aircraft models, the camera model and the number of pixels in each dimension of the image (512 × 512 in these examples) were the only pieces of world knowledge input to ACRONYM. It has no special knowledge of aerial scenes: all its rules are about geometry and algebraic manipulation. These were applied to the particular generic models it was given, to make predictions and then to carry out interpretations.

Figures 3.3 through 3.5 show three examples of interpretations carried out by ACRONYM. In each case part *a* is a half-tone of the original grey level image. The *b* version is the result of applying the line finder of Nevatia and Babu [8]. That line finder was designed to find linear features such as roads and rivers in aerial photos. Close examination of results on these images indicate many errors, and undue enlargement in width of narrow linear features. It also produces many noise edges in in smooth brightness gradients (not visible at the resolution of the reproductions of these figures). These edges are the lowest level input to ACRONYM.

An edge linker [4] is directed by the predictions to look for ribbons and ellipses. In this case there is very little a *priori* information about the scale of the images. The *c* versions of each figure show the ribbons fitted to the linked edges when it is searching for candidate matches for the fuselage and wings of aircraft. There is even further degredation of image information at this stage. This is the only data which the ACRONYM reasoning system is given to interpret. Notice that in the figure 3.5 almost all the shapes corresponding to aircraft are lost. Quite a few aircraft in 3.3 are lost also. Besides losing many shapes, the combination of the edge finder and edge linker conspire to give very inaccurate image measurements. We assume all image measurements have a ±30% error, except that for very small measurements, we assume that pixel noise swamps even those error estimates. Then the error is estimated to be inversely proportional to the measurement with a 2 pixel measurement admitting a 100% error. Thus the data which ACRONYM really gets to work with is considerably more fuzzy than indicated by the the c series of figures.

We intend to make use of new and better low level descriptive processes being developped in our laboratory by other researchers as soon as they become robust enough for every day use (e.g. Baker [1] whose descriptions from stereo will also include surface and depth information).

Despite this very noisy descriptive data ACRONYM makes good interpretations of the images. The *d* series of figures show its interpretations with the ribbons labeled by what part of the model they were matched to. (The numbers which may be unreadable in 3.3*d* show the groupings into individual aircraft.)

ACRONYM first uses the most general set of constraints, those associated with the generic class of wide-bodied jets, when carrying out intitial prediction and interpretation. Interpretation adds additional constraints for each hypothesized aircraft instance. For example in finding the correspondences in figure 3.4*d* constraints were added which eventually constrained the WING_WIDTH (the width of the wings where they attach to fuselage) to lie in the range [7, 10.5677531] compared to the modeled bounds of [7, 12]. The height of the camera, modeled to lie in the range [1000, 12000] is constrained by the interpretation to the range [2199, 3322].

Once a consistent match or partial match to a geometric model has been found in the context of some set of constraints (model class), it easy to check whether it might also be an instance of a subclass. We need only add the extra constraints associated with the subclass and check for consistency with those already implied by the interpretation using the CMS as described in section 2.1. The aircraft located in 3.4*d* is consistent with the constraints for an L-1011, but not for a Boeing-747. Author examination of the images had previously indicated that the aircraft was an L-1011. The additional symbolic constraints implied by accepting that the aircraft is in fact an L-1011 propagate through the entire constraint set. Although the constraints decribing an L-1011 do not include constraints on camera height, the back constraints deduced during interpretation relate quantifers representing such quantities as length of the wings to the height (and focal ratio in the more general case). Thus the height of the camera is further constrained in 3.4*d* to lie in the range [2356, 2489]. Recall that all image measurements were subject to ±30% errors, and that this estimate has taken all such errors into account.

Figure 3.3*d* indicates matches were found for three airplanes. Examination of the data in 3.3*c* indicates that this is the best that could be expected. Not' however that only partial matches were found in all three cases. For such small ribbons errors were apparently larger than the generous estimate used. The fuselage ribbon in the leftmost aircraft (number 1) for instance fails to pass the coarse filtering stage. Despite the partial match, this particular aircraft is found to be consistent with the constraints for an L-1011, but not consistent with those of a Boeing-747. Again this is correct.

The other two aircraft identified are even more interesting. The author had thought from casual inspection of the grey level image that they were instances of Boeing-747s. They both gave matches consistent with the class of wide-bodied jets. As expected neither was consistent with the extra costraints of an L-1011. However, although each individual parameter range from the interpretation constraint sets was consistent with the individual parameter value or range for the class of Bᴇeing-747s, neither set of constraints was consistent with that subclass (the constraints contain much finer information than just the parameter ranges - in the same many as in the example above where constraints on wing length propagate to constrain the camera height). On close examination of the grey-level image it was determined that the aircraft were not in fact Boeing-747's. The author used the fact that they were much smaller than the L-1011 to make that deduction, but the system made the deduction at the local level before considering comparisons between aircraft.

The aircraft (probably Boeing-707's, but at the time of writing we haven't yet got engineering drawings needed to build an accurate model for ACRONYM to check against the images) are in fact too small to be wide-bodied jets of any type. Since the scale of the image is unknown *a priori* this can not be deduced locally. However it is reflected in the height estimates derived at the local level — [5400, 8226] interpreting the L-1011 just as a generic wide-body, ([5786, 6170] as an L-1011), and [9007, 11846] for the rightmost aircraft. Thus ACRONYM deduces that either the left aircraft is a wide-body and the others are not, or the right two are wide-bodies and the left one is not (it is too big).

Finally note that geometrically there were other candidates for aircraft in the ribbons of figure 3.3c. For instance the wing of the aircraft just to the right of those indentified and a ribbon found for its passenger ramp could be the two wings of an aircaft with a fuselage missing between them. In fact these two ribbons were instantiated as an aircaft on the basis of the coarse filters on the nodes and arcs. However the set of back constraints they generated were mutually inconsistent.

Thus we can see from the examples that even with very poor and noisy data the combined use of geonetry and symbolic algebraic constraints can lead to accurate image interpretations. The system should be tested on more accurate low level data to fully evaluate the power of this approach.

```
ENG-DISP-GAP ϵ [6, 10]
ENG-DISP ϵ [0, 4]
ENG-GAP ϵ [7, 10]
STAB-ATTACH ϵ [3, 5]
R-ENG-ATTACHMENT ϵ [3, 5]
ENG-OUT ϵ [5, 12]
WING-ATTACHMENT ϵ [20, 40]
        WING-ATTACHMENT ≥ 0.4*FUSELAGE-LENGTH
        WING-ATTACHMENT ≤ 0.6*FUSELAGE-LENGTH
STAB-RATIO ϵ [0.2, 0.55]
STAB-SWEEP-BACK ϵ [3, 7]
STAB-LENGTH ϵ [7.6, 13]
STAB-THICK ϵ [0.7, 1.1]
STAB-WIDTH ϵ [5, 11]
RUDDER-RATIO ϵ [0.3, 0.4]
RUDDER-SWEEP-BACK ϵ [3, 9]
RUDDER-LENGTH ϵ [8.5, 14.2]
RUDDER-X-HEIGHT ϵ [7, 13]
RUDDER-X-WIDTH ϵ [0.7, 1.1]
WING-RATIO ϵ [0.35, 0.45]
WING-THICK ϵ [1.5, 2.5]
WING-WIDTH ϵ [7, 12]
        WING-WIDTH ≤ 0.5*WING-LENGTH
WING-LIFT ϵ [1, 2]
WING-SWEEP-BACK ϵ [10, 18]
WING-LENGTH ϵ [22, 33.5]
        WING-LENGTH ≥ 2*WING-WIDTH
        WING-LENGTH ≥ 0.43*FUSELAGE-LENGTH
        WING-LENGTH ≤ 0.65*FUSELAGE-LENGTH
REAR-ENGINE-LENGTH ϵ [6, 10]
ENGINE-LENGTH ϵ [4, 7]
ENGINE-RADIUS ϵ [1, 1.8]
FUSELAGE-RADIUS ϵ [2.5, 4]
FUSELAGE-LENGTH ϵ [40, 70]
        FUSELAGE-LENGTH ≥ 1.66666666*WING-ATTACHMENT
        FUSELAGE-LENGTH ≥ 1.53846154*WING-LENGTH
        FUSELAGE-LENGTH ≤ 2.5*WING-ATTACHMENT
        FUSELAGE-LENGTH ≤ 2.3255814*WING-LENGTH
R-ENG-QUANT ϵ [0, 1]
        R-ENG-QUANT ≤ 2 + -1*F-ENG-QUANT
F-ENG-QUANT ϵ [1, 2]
        F-ENG-QUANT ≤ 2 + -1*R-ENG-QUANT
```

Fig. 3.2

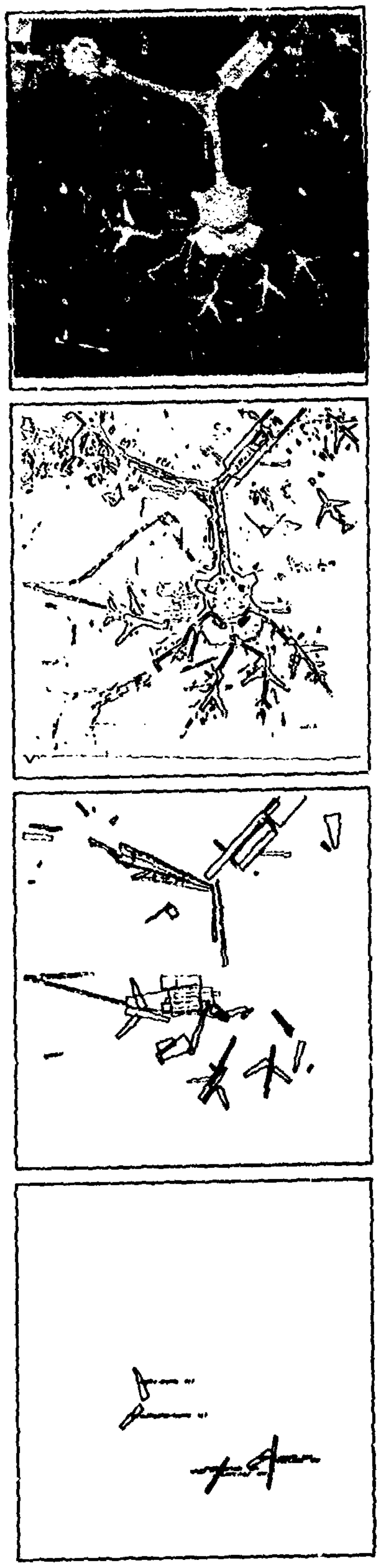


Fig. 3.3 a,b,c,d

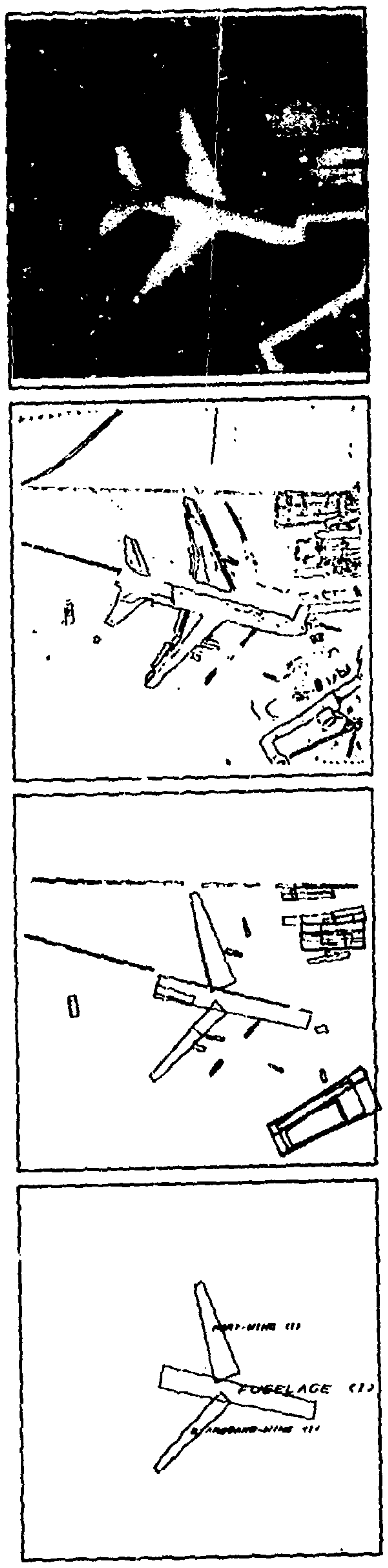


Fig. 3.4 a,b,c,d

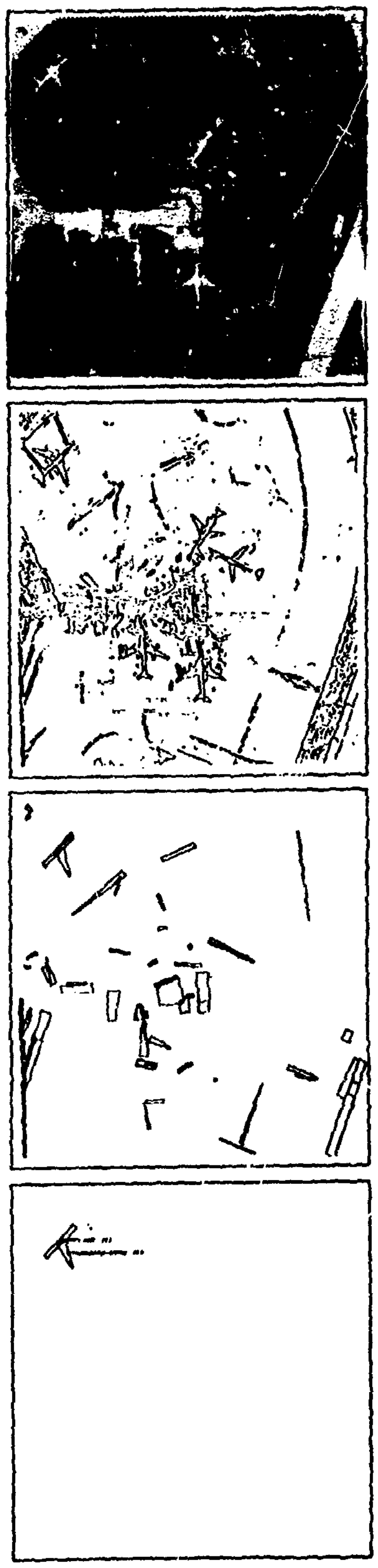


Fig. 3.5 a,b,c,d

References

[1] Baker, H. Harlyn; *Edge Based Stereo Correlation*, Proceedings ARPA Image Understanding Workshop, College Park, MD, April 1980, 168-175.

[2] Bledsoe, W. W.; *The Sup-Inf Method in Presburger Arithmetic*, Memo ATP-18, Dept. of Math. and Comp. Sci., University of Texas at Austin, Austin, Texas, Dec. 1974.

[3] Bobrow, Daniel G. and Terry Winograd; *An Overview of KRL, a Knowledge Representation Language*, Cognitive Science 1, 1977, 3-46.

[4] Brooks, Rodney A.; *Goal-Directed Edge Linking and Ribbon Finding*, Proceedings ARPA Image Understanding Workshop, Menlo Park, April 1979, 72-76.

[5] Brooks, Rodney A.; *Symbolic Reasoning Among 3-D Models and 2-D Images*, AI Journal, to appear.

[6] Brooks, Rodney A. and Thomas O. Binford; *Representing and Reasoning About Partially Specified Scenes*, Proceedings ARPA Image Understanding Workshop, College Park, MD, April 1980, 95-103.

[7] Brooks, Rodney A. and Thomas O. Binford; *Interpretive Vision and Restriction Graphs*, Proceedings of the First Annual National Conference on Artificial Intelligence, Stanford, August 1980, 21-27.

[8] Nevatia, Ramakant and K. Ramesh Babu; *Linear Feature Extraction and Description*, Computer Graphics and Image Processing 13, 1980, 257-269.

[9] Shostak, Robert E.; *On the SUP-INF Method for Proving Presburger Formulas*, JACM 24, 1977, 529-543.

Determining Optical Flow

Berthold K. P. Horn and Brian G. Schunck

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square, NE43-811
Cambridge, Massachusetts 02139

Abstract

Optical flow cannot be computed locally, since only one independent measurement is available from the image sequence at a point, while the flow velocity has two components. A second constraint is needed. A method for finding the optical flow pattern is presented which assumes that the apparent velocity of the brightness pattern varies smoothly almost everywhere in the image. An iterative implementation is shown which successfully computes the optical flow for a number of synthetic image sequences. The algorithm is robust in that it can handle image sequences that are quantized rather coarsely in space and time. It is also insensitive to quantization of brightness levels and additive noise. Examples are included where the assumption of smoothness is violated at singular points or along lines in the image.

1. Introduction

Optical flow is the distribution of apparent velocities of movement of brightness patterns in an image. Optical flow can arise from relative motion of objects and the viewer [8, 9]. Consequently, optical flow can give important information about the spatial arrangement of the objects viewed and the rate of change of this arrangement [10]. Discontinuities in the optical flow can help in segmenting images into regions that correspond to different objects [29]. Attempts have been made to perform such segmentation using differences between successive image frames [17, 18, 19, 22, 3, 27]. Several papers address the problem of recovering the motions of objects relative to the viewer from the optical flow [12, 20, 21, 23, 31]. Some recent papers provide a clear exposition of this enterprise [32, 33]. The mathematics can be made rather difficult, by the way, by chosing an inconvenient coordinate system. In some cases information about the shape of an object may also be recovered [4, 20, 21].

These papers begin by assuming that the optical flow has already been determined. Although some reference has been made to schemes for computing the flow from successive views of a scene [7, 12], the specifics of a scheme for determining the flow from the image have not been described. Related work has been done in an attempt to formulate a model for the short range motion detection processes in human vision [2, 24]. The pixel recursive equations of Netravali and Robbins [30], designed for coding motion in television signals, bear some similarity to the iterative equations developed in this paper. A recent review [28] of computational techniques for the analysis of image sequences contains over 150 references.

The optical flow cannot be computed at a point in the image independently of neighboring points without introducing additional constraints, because the velocity field at each image point has two components while the change in image brightness at a point in the image plane due to motion yields only one constraint. Consider, for example, a patch of a pattern where brightness[1] varies as a function of one image coordinate but not the other. Movement of the pattern in one direction alters the brightness at a particular point, but motion in the other direction yields no change. Thus components of movement in the latter direction cannot be determined locally.

2. Relationship to Object Motion

The relationship between the optical flow in the image plane and the velocities of objects in the three dimensional world is not necessarily obvious. We perceive motion when a changing picture is projected onto a stationary screen, for example. Conversely, a moving object may give rise to a constant brightness pattern. Consider, for example, a uniform sphere which exhibits shading because its surface elements are oriented in many different directions. Yet, when it is rotated, the optical flow is zero at all points in the image, since the shading does not move with the surface. Also, specular reflections move with a velocity characteristic of the virtual image, not the surface in which light is reflected.

For convenience, we tackle a particularly simple world where the apparent velocity of brightness patterns can be directly identified with the movement of surfaces in the scene.

3. The Restricted Problem Domain

To avoid variations in brightness due to shading effects we initially assume that the surface being imaged is flat. We further assume that the incident illumination is uniform across the surface. The brightness

[1] In this paper, the term brightness means image irradiance. The brightness pattern is the distribution of irradiance in the image.

at a point in the image is then proportional to the reflectance of the surface at the corresponding point on the object. Also, we assume at first that reflectance varies smoothly and has no spatial discontinuities. This latter condition assures us that the image brightness is differentiable. We exclude situations where objects occlude one another, in part, because discontinuities in reflectance are found at object boundaries. In two of the experiments discussed later, some of the problems occasioned by occluding edges are exposed.

In the simple situation described, the motion of the brightness patterns in the image is determined directly by the motions of corresponding points on the surface of the object. Computing the velocities of points on the object is a matter of simple geometry once the optical flow is known.

4. Constraints

We will derive an equation that relates the change in image brightness at a point to the motion of the brightness pattern. Let the image brightness at the point (x, y) in the image plane at time t be denoted by $E(x, y, t)$. Now consider what happens when the pattern moves. The brightness of a particular point in the pattern is constant, so that

$$\frac{dE}{dt} = 0$$

Using the chain rule for differentiation we see that,

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0$$

(See Appendix A for a more detailed derivation.) If we let

$$u = \frac{dx}{dt} \quad \text{and} \quad v = \frac{dy}{dt},$$

then it is easy to see that we have a single linear equation in the two unknowns u and v,

$$E_x u + E_y v + E_t = 0,$$

where we have also introduced the additional abbreviations E_x, E_y, and E_t for the partial derivatives of image brightness with respect to x, y and t, respectively. The constraint on the local flow velocity expressed by this equation is illustrated in Figure 1. Writing the equation in still another way,

$$(E_x, E_y) \cdot (u, v) = -E_t.$$

Thus the component of the movement in the direction of the brightness gradient (E_x, E_y) equals

$$-\frac{E_t}{\sqrt{E_x^2 + E_y^2}}.$$

We cannot, however determine the component of the movement in the direction of the iso-brightness contours, at right angles to the brightness gradient. As a consequence, the flow velocity (u, v) cannot be computed locally without introducing additional constraints.

5. The Smoothness Constraint

If every point of the brightness pattern can move independently, there is little hope of recovering the velocities. More commonly we view opaque objects of finite size undergoing rigid motion or deformation. In this case neighboring points on the objects have similar velocities and the velocity field of the brightness patterns in the image varies smoothly almost everywhere. Discontinuities in flow can be expected where one object occludes another. An algorithm based on a smoothness constraint is likely to have difficulties with occluding edges as a result.

One way to express the additional constraint is to minimize the square of the magnitude of the gradient of the optical flow velocity:

$$\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \quad \text{and} \quad \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2.$$

Another measure of the smoothness of the optical flow field is the sum of the squares of the Laplacians of the two velocity components. The Laplacians of u and v are defined as

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \quad \text{and} \quad \nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}.$$

In simple situations, both Laplacians are zero. If the viewer translates parallel to a flat object, rotates about a line perpendicular to the surface or travels orthogonally to the surface, then the second partial derivatives of both u and v vanish (assuming perspective projection in the image formation.)

In this paper, we will use the square of the magnitide of the gradient as our smoothness measure. Note that our approach is in contrast with that of [7], who propose an algorithm that incorporates additional assumptions such as constant flow velocities within discrete regions of the image. Their method, based on cluster analysis, cannot deal with rotating objects, since these give rise to a continuum of flow velocities.

6. Quantization and Noise

Images may be sampled at intervals on a fixed grid of points. While tesselations other than the obvious one have certain advantages [11, 12], for convenience we will assume that the image is sampled on a square grid at regular intervals. Let the measured brightness be $E_{i,j,k}$ at the intersection of the i-th row and j-th column in the k-th image frame. Ideally, each measurement should be an average over the area of a picture cell and over the length of the time interval. In the experiments cited here we have taken samples at discrete points in space and time instead.

In addition to being quantized in space and time, the measurements will in practice be quantized in brightness as well. Further, noise will be apparent in measurements obtained in any real system.

7. Estimating the Partial Derivatives

We must estimate the derivatives of brightness from the discrete set of image brightness measurements available. It is important that the estimates of E_x, E_y, and E_t be consistent. That is, they should all re[illegible] me point in the image at the same time. While there are [illegible] formulas for approximate differentiation [5, 13] we will use a [illegible] which gives us an estimate of E_x, E_y, E_t at a point in the center of a cube formed by eight measurements. The relationship in space and time between these measurements is shown in Figure 2. Each of the estimates is the average of four first differences taken over adjacent measurements in the cube.

$$\begin{aligned}
E_x \approx \frac{1}{4}\{&E_{i,j+1,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i+1,j,k} \\
&+ E_{i,j+1,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i+1,j,k+1}\} \\
E_y \approx \frac{1}{4}\{&E_{i+1,j,k} - E_{i,j,k} + E_{i+1,j+1,k} - E_{i,j+1,k} \\
&+ E_{i+1,j,k+1} - E_{i,j,k+1} + E_{i+1,j+1,k+1} - E_{i,j+1,k+1}\} \\
E_t \approx \frac{1}{4}\{&E_{i,j,k+1} - E_{i,j,k} + E_{i+1,j,k+1} - E_{i+1,j,k} \\
&+ E_{i,j+1,k+1} - E_{i,j+1,k} + E_{i+1,j+1,k+1} - E_{i+1,j+1,k}\}.
\end{aligned}$$

Here the unit of length is the grid spacing interval in each image frame and the unit of time is the image frame sampling period. We avoid estimation formulae with larger support, since these typically are equivalent to formulae of small support applied to smoothed images [16].

8. Estimating the Laplacian of the Flow Velocities

As will be shown in the next section, we will also need to approximate the Laplacians of u and v. One convenient approximation takes the following form

$$\nabla^2 u \approx \kappa(\bar{u}_{i,j,k} - u_{i,j,k}) \quad \text{and} \quad \nabla^2 v \approx \kappa(\bar{v}_{i,j,k} - v_{i,j,k}),$$

where the local averages $\bar{u}$ and $\bar{v}$ are defined as follows

$$\begin{aligned}
\bar{u}_{i,j,k} &= \frac{1}{6}\{u_{i-1,j,k} + u_{i,j+1,k} + u_{i+1,j,k} + u_{i,j-1,k}\} \\
&+ \frac{1}{12}\{u_{i-1,j-1,k} + u_{i-1,j+1,k} + u_{i+1,j+1,k} + u_{i+1,j-1,k}\} \\
\bar{v}_{i,j,k} &= \frac{1}{6}\{v_{i-1,j,k} + v_{i,j+1,k} + v_{i+1,j,k} + v_{i,j-1,k}\} \\
&+ \frac{1}{12}\{v_{i-1,j-1,k} + v_{i-1,j+1,k} + v_{i+1,j+1,k} + v_{i+1,j-1,k}\}.
\end{aligned}$$

The proportionality factor κ equals 3 if the average is computed as shown and we again assume that the unit of length equals the grid spacing interval. Figure 3 illustrates the assignment of weights to neighboring points. The approximation for the Laplacian using the center cell and all eight neighbors is more stable than the usual one based on the center cell and its four horizontal and vertical neighbors only.

9. Minimization

The problem then is to minimize the sum of the errors in the equation for the rate of change of image brightness,

$$\mathcal{E}_b = E_x u + E_y v + E_t,$$

and the measure of the departure from smoothness in the velocity flow,

$$\mathcal{E}_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

What should be the relative weight of these two factors? In practice the image brightness measurements will be corrupted by quantization error and noise so that we cannot expect $\mathcal{E}_b$ to be identically zero. This quantity will tend to have an error magnitude that is proportional to the noise in the measurement. This fact guides us in choosing a suitable weighting factor, denoted by α^2, as will be seen later.

Let the total error to be minimized be

$$\mathcal{E}^2 = \int\!\!\int \alpha^2 \mathcal{E}_c^2 + \mathcal{E}_b^2 \, dx\, dy.$$

The minimization is to be accomplished by finding suitable values for the optical flow velocity (u, v). Using the calculus of variations [6, pp. 191–92], we obtain

$$\begin{aligned}
E_x^2 u + E_x E_y v &= \alpha^2 \nabla^2 u - E_x E_t \\
E_x E_y u + E_y^2 v &= \alpha^2 \nabla^2 v - E_y E_t.
\end{aligned}$$

Using the approximation to the Laplacian introduced in the previous section,

$$\begin{aligned}
(\alpha^2 + E_x^2)u + E_x E_y v &= (\alpha^2 \bar{u} - E_x E_t) \\
E_x E_y u + (\alpha^2 + E_y^2)v &= (\alpha^2 \bar{v} - E_y E_t).
\end{aligned}$$

The determinant of the coefficient matrix equals $\alpha^2(\alpha^2 + E_x^2 + E_y^2)$. Solving for u and v we find that

$$\begin{aligned}
(\alpha^2 + E_x^2 + E_y^2)u &= +(\alpha^2 + E_y^2)\bar{u} - E_x E_y \bar{v} - E_x E_t \\
(\alpha^2 + E_x^2 + E_y^2)v &= -E_x E_y \bar{u} + (\alpha^2 + E_x^2)\bar{v} - E_y E_t.
\end{aligned}$$

10. Difference of Flow at a Point from Local Average

These equations can be written in the alternate form

$$\begin{aligned}
(\alpha^2 + E_x^2 + E_y^2)(u - \bar{u}) &= -E_x[E_x \bar{u} + E_y \bar{v} + E_t] \\
(\alpha^2 + E_x^2 + E_y^2)(v - \bar{v}) &= -E_y[E_x \bar{u} + E_y \bar{v} + E_t].
\end{aligned}$$

This shows that the value of the flow velocity (u, v) which minimizes the error $\mathcal{E}^2$ lies in the direction towards the constraint line along a line that intersects the constraint line at right angles. This relationship is illustrated geometrically in Figure 4. The distance from the local average is proportional to the error in the basic formula for rate of change of brightness when $\bar{u}$, $\bar{v}$ are substituted for u and v. Finally we can see that α^2 plays a significant role only for areas where the brightness gradient is small, preventing haphazard adjustments to the estimated flow velocity occasioned by noise in the estimated derivatives. This parameter should be roughly equal to the expected noise in the estimate of $E_x^2 + E_y^2$.

11. Constrained Minimization

When we allow α^2 to tend to zero we obtain the solution to a constrained minimization problem. Applying the method of Lagrange multipliers [35, 36] to the problem of minimizing $\mathcal{E}_c^2$ while maintaining $\mathcal{E}_b = 0$ leads to

$$E_y\nabla^2 u = E_x\nabla^2 v, E_x u + E_y v + E_t = 0.$$

Approximating the Laplacian by the difference of the velocity at a point and the average of its neighbors then gives us

$$(E_x^2 + E_y^2)(u - \bar{u}) = -E_x[E_x\bar{u} + E_y\bar{v} + E_t]$$
$$(E_x^2 + E_y^2)(v - \bar{v}) = -E_y[E_x\bar{u} + E_y\bar{v} + E_t].$$

Referring again to Figure 4, we note that the point computed here lies at the intersection of the constraint line and the line at right angles through the point $(\bar{u}, \bar{v})$. We will not use these equations since we do expect errors in the estimation of the partial derivatives.

12. Iterative Solution

We now have a pair of equations for each point in the image. It would be very costly to solve these equations simultaneously by one of the standard methods, such as Gauss-Jordan elimination [13, 14]. The corresponding matrix is sparse and very large since the number of rows and columns equals twice the number of picture cells in the image. Iterative methods, such as the Gauss-Seidel method [13, 15], suggest themselves. We can compute a new set of velocity estimates (u^{n+1}, v^{n+1}) from the estimated derivatives and the average of the previous velocity estimates (u^n, v^n) by

$$u^{n+1} = \bar{u}^n - E_x[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2)$$
$$v^{n+1} = \bar{v}^n - E_y[E_x\bar{u}^n + E_y\bar{v}^n + E_t]/(\alpha^2 + E_x^2 + E_y^2).$$

(It is interesting to note that the new estimates at a particular point do not depend directly on the previous estimates at the same point.)

The natural boundary condition for the variational problem turns out to be a zero normal derivative. At the edge of the image, some of the points needed to compute the local average of velocity lie outside the image. Here we simply copy velocities from adjacent points further in.

13. Filling In Uniform Regions

In parts of the image where the brightness gradient is zero, the velocity estimates will simply be averages of the neighboring velocity estimates. There is no local information to constrain the apparent velocity of motion of the brightness pattern in these areas. Eventually the values around such a region will propagate inwards. If the velocities on the border of the region are all equal to the same value, then points in the region will be assigned that value too, after a sufficient number of iterations. Velocity information is thus filled in from the boundary of a region of constant brightness.

If the values on the border are not all the same, it is a little more difficult to predict what will happen. In all cases, the values filled in will correspond to the solution of the Laplace equation for the given boundary condition [1, 26, 34].

The progress of this filling-in phenomena is similar to the propagation effects in the solution of the heat equation for a uniform flat plate, where the time rate of change of temperature is proportional to the Laplacian. This gives us a means of understanding the iterative method in physical terms and of estimating the number of steps required. The number of iterations should be larger than the number of pictures cells across the largest region that must be filled in. If the size of such regions is not known in advance one may use the cross-section of the whole image as a conservative estimate.

14. Tightness of Constraint

When brightness in a region is a linear function of the image coordinates we can only obtain the component of optical flow in the direction of the gradient. The component at right angles is filled in from the boundary of the region as described before. In general the solution is most accurately determined in regions where the brightness gradient is not too small and varies in direction from point to point. Information which constrains both components of the optical flow velocity is then available in a relatively small neighborhood. Too violent fluctuations in brightness on the other hand are not desirable since the estimates of the derivatives will be corrupted as the result of undersampling and aliasing.

15. Choice of Iterative Scheme

As a practical matter one has a choice of how to interlace the iterations with the time steps. On the one hand, one could iterate until the solution has stabilized before advancing to the next image frame. On the other hand, given a good initial guess one may need only one iteration per time-step. A good initial guess for the optical flow velocities is usually available from the previous time-step.

The advantages of the latter approach include an ability to deal with more images per unit time and better estimates of optical flow velocities in certain regions. Areas in which the brightness gradient is small lead to uncertain, noisy estimates obtained partly by filling in from the surround. These estimates are improved by considering further images. The noise in measurements of the images will be independent and tend to cancel out. Perhaps more importantly, different parts of the pattern will drift by a given point in the image. The direction of the brightness gradient will vary with time, providing information about both components of the optical flow velocity.

A practical implementation would most likely employ one iteration per time step for these reasons. We illustrate both approaches in the experiments.

16. Experiments

The iterative scheme has been implemented and applied to image sequences corresponding to a number of simple flow patterns. The results shown here are for a relatively low resolution image of 32 by 32 picture cells. The brightness measurements were intentionally corrupted by approximately 1% noise and then quantized into 256 levels to simulate a real imaging situation. The underlying surface reflectance pattern was a linear combination of spatially orthogonal sinusoids. Their wavelength was chosen to give reasonably strong brightness gradients without leading to undersampling problems. Discontinuities were avoided to ensure that the required derivatives exist everywhere.

Shown in Figure 5, for example, are four frames of a sequence of images depicting a sphere rotating about an axis inclined towards the viewer. A smoothly varying reflectance pattern is painted on the surface of the sphere. The sphere is illuminated uniformly from all directions so that there is no shading. We chose to work with synthetic image sequences so that we can compare the results of the optical flow computation with the exact values calculated using the transformation equations relating image coordinates to coordinates on the underlying surface reflectance pattern.

17. Results

The first flow to be investigated was a simple linear translation of the entire brightness pattern. The resulting computed flow is shown as a needle diagram in Figure 6 for 1, 4, 16, and 64 iterations. The estimated flow velocities are depicted as short lines, showing the apparent displacement during one time step. In this example a single time step was taken so that the computations are based on just two images. Initially the estimates of flow velocity are zero. Consequently the first iteration shows vectors in the direction of the brightness gradient. Later, the estimates approach the correct values in all parts of the image. Few changes occur after 32 iterations when the velocity vectors have errors of about 10%. The estimates tend to be too small, rather than too large, perhaps because of a tendency to underestimate the derivatives. The worst errors occur, as one might expect, where the brightness gradient is small.

In the second experiment one iteration was used per time step on the same linear translation image sequence. The resulting computed flow is shown in Figure 7 for 1, 4, 16, and 64 time steps. The estimates approach the correct values more rapidly and do not have a tendency to be too small, as in the previous experiment. Few changes occur after 16 iterations when the velocity vectors have errors of about 7%. The worst errors occur, as one might expect, where the noise in recent measurements of brightness was worst. While individual estimates of velocity may not be very accurate, the average over the whole image was within 1% of the correct value.

Next, the method was applied to simple rotation and simple contraction of the brightness pattern. The results after 32 time steps are shown in Figure 8. Note that the magnitude of the velocity is proportional to the distance from the origin of the flow in both of these cases. (By origin we mean the point in the image where the velocity is zero.)

In the examples so far the Laplacian of both flow velocity components is zero everywhere. We also studied more difficult cases where this was not the case. In particular, if we let the magnitude of the velocity vary as the inverse of the distance from the origin we generate flow around a line vortex and two dimensional flow into a sink. The computed flow patterns are shown in Figure 9. In these examples, the computation involved many iterations based on a single time step. The worst errors occur near the singularity at the origin of the flow pattern, where velocities are found which are much larger than one picture cell per time step.

Finally we considered rigid body motions. Shown in Figure 10 are the flows computed for a cylinder rotating about its axis and for a rotating sphere. In both cases the Laplacian of the flow is not zero and in fact the Laplacian of one of the velocity components becomes infinite on the occluding bound. Since the velocities themselves remain finite, reasonable solutions are still obtained. The correct flow patterns are shown in Figure 11. Comparing the computed and exact values shows that the worst errors occur on the occluding boundary. These boundaries constitute a one dimensional subset of the plane and so one can expect that the relative number of points at which the estimated flow is seriously in error will decrease as the resolution of the image is made finer.

In Appendix B it is shown that there is a direct relationship between the Laplacian of the flow velocity components and the Laplacian of the surface height. This can be used to see how our smoothness constraint will fare for different objects. For example, a rotating polyhedron will give rise to a flow which has zero Laplacian except on the image lines which are the projections of the edges of the body.

18. Summary

A method has been developed for computing optical flow from a sequence of images. It is based on the observation that the flow velocity has two components and that the basic equation for the rate of change of image brightness provides only one constraint. Smoothness of the flow was introduced as a second constraint. An iterative method for solving the resulting equation was then developed. A simple implementation provided visual confirmation of convergence of the solution in the form of needle diagrams. Examples of several different types of optical flow patterns were studied. These included cases where the Laplacian of the flow was zero as well as cases where it became infinite at singular points or along bounding curves.

The computed optical flow is somewhat inaccurate since it is based on noisy, quantized measurements. Proposed methods for obtaining information about the shapes of objects using derivatives (divergence and curl) of the optical flow field may turn out to be impractical since the inaccuracies will be amplified.

19. Acknowledgement

This research was conducted at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-75-C-0643. One of the authors (Horn) would like to thank Professor H.-H. Nagel for his hospitality. The basic equations were conceived during a visit to the University of Hamburg, stimulated by Professor Nagel's long-standing interest in motion vision. The other author (Schunck) would like to thank W. E. L. Grimson and E. Hildreth for many interesting discussions and much knowledgable criticism. W. E. L. Grimson and Katsushi Ikeuchi helped to illuminate a conceptual bug in an earlier version of this paper. We would also like to thank J. Jones for preparing the drawings.

Appendix A. Rate of Change of Image Brightness

Consider a patch of the brightness pattern that is displaced a distance δx in the x-direction and δy in the y-direction in time δt. The brightness of the patch is assumed to remain constant so that

$$E(x,y,t) = E(x+\delta x, y+\delta y, t+\delta t).$$

Expanding the right hand side about the point (x,y,t) we get,

$$E(x,y,t) = E(x,y,t) + \delta x\frac{\partial E}{\partial x} + \delta y\frac{\partial E}{\partial y} + \delta t\frac{\partial E}{\partial t} + \epsilon.$$

Where ϵ contains second and higher order terms in δx, δy, and δt. After subtracting $E(x,y,t)$ from both sides and dividing through by δt we have

$$\frac{\delta x}{\delta t}\frac{\partial E}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} + O(\delta t) = 0,$$

where $O(\delta t)$ is a term of order δt (we assume that δx and δy vary as δt.) In the limit as $\delta t \to 0$ this becomes

$$\frac{\partial E}{\partial x}\frac{dx}{dt} + \frac{\partial E}{\partial y}\frac{dy}{dt} + \frac{\partial E}{\partial t} = 0.$$

Appendix B. Smoothness of Flow for Rigid Body Motions

Let a rigid body rotate about an axis $(\omega_x, \omega_y, \omega_z)$, where the magnitude of the vector equals the angular velocity of the motion. If this axis passes through the origin, then the velocity of a point (x, y, z) equals the cross product of $(\omega_x, \omega_y, \omega_z)$, and (x, y, z). There is a direct relationship between the image coordinates and the x and y coordinates here if we assume that the image is generated by orthographic projection. The x and y components of the velocity can be written,

$$u = \omega_y z - \omega_z y$$
$$v = \omega_z x - \omega_x z.$$

Consequently,

$$\nabla^2 u = +\omega_y \nabla^2 z$$
$$\nabla^2 v = -\omega_x \nabla^2 z.$$

This illustrates that the smoothness of the optical flow is related directly to the smoothness of the rotating body and that the Laplacian of the flow velocity will become infinite on the occluding bound, since the partial derivatives of z with respect to x and y become infinite there.

References

1. W. F. Ames, *Numerical Methods for Partial Differential Equations*. New York: Academic Press, 1977.

2. J. Batali and S. Ullman, "Motion detection and analysis," in *Proc. ARPA Image Understanding Workshop*, pp 69-75, November 1979.

3. C. Cafforio and F. Rocca, "Methods for measuring small displacements of television images," *IEEE Trans. Information Theory*, vol. IT-22, pp. 573-579, September 1976.

4. W. Clocksin, "Determining the orientation of surfaces from optical flow," in *Proc. 3rd AISB Conference*, Hamburg, pp. 93-102.

5. S. D. Conte and C. de Boor, *Elementary Numerical Analysis*. New York: McGraw-Hill, 1972.

6. R. Courant and D. Hilbert, *Methods of Mathematical Physics*. Vol. 1. New York: John Wiley & Sons, 1937.

7. C. L. Fennema and W. B. Thompson, "Velocity determination in scenes containing several moving objects," *Computer Graphics and Image Processing*, vol. 9, pp. 301-315, April 1979.

8. J. J. Gibson, *The Perception of the Visual World*. Cambridge, England: Riverside Press, 1950.

9. J. J. Gibson, *The Senses Considered as Perceptual Systems*. Boston: Houghton-Mifflin, 1966.

10. J. J. Gibson, "On the analysis of change in the optic array," *Scandinavian J. Psych.*, vol. 18, pp. 161-163, 1977.

11. S. B. Gray, "Local properties of binary images in two dimensions," *IEEE Trans. Computers*, vol. C-20, pp. 551-561, May 1971.

12. Y. Hadani, G. Ishai, and M. Gur, "Visual stability and space perception in monocular vision: Mathematical model," *J. Opt. Soc. Amer.*, vol. 70, pp. 60-65, January 1980.

13. R. W. Hamming, *Numerical Methods for Scientists and Engineers*. New York: McGraw-Hill, 1962.

14. F. B. Hildebrand, *Methods of Applied Mathematics*. Englewood Cliffs, N. J.: Prentice-Hall, 1965.

15. F. B. Hildebrand, *Introduction to Numerical Analysis*. New York: McGraw-Hill, 1974.

16. B. K. P. Horn, "Hill shading and the reflectance map," *Proc. IEEE*, vol. 69, pp. 14-47, January 1981.

17. R. Jain, W. N. Martin, and J. K. Aggarwal "Segmentation through the detection of changes due to motion," *Computer Graphics and Image Processing*, vol. 11, pp. 13-34, September 1979.

18. R. Jain, D. Militzer, and H.-H. Nagel, "Seperating non-stationary from stationary scene components in a sequence of real world TV-images," in *Proc. 5th Int. Joint Conf. Artificial Intelligence*, pp. 612-618, August 1977.

19. R. Jain and H.-H. Nagel, "On the analysis of accumulative difference pictures from image sequences of real world scenes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-1, pp. 206-214, April 1979.

20. J. J. Koenderink and A. J. van Doorn, "Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer," *Optica Acta*, vol. 22, pp. 773-791, 1975.

21. J. J. Koenderink and A. J. van Doorn, "Visual perception of rigidity of solid shape," *J. Math. Bio.*, vol. 3, pp. 79-85, 1976.

22. J. O. Limb and J. A. Murphy, "Estimating the velocity of moving images in television signals," *Computer Graphics and Image Processing*, vol. 4, pp. 311-327, December 1975.

23. H. C. Longuet-Higgins and K. Prazdny, "The interpretation of moving retinal image," *Proc. R. Soc. B*, vol. 208, pp. 385-397, 1980.

24. D. Marr and S. Ullman, "Directional selectivity and its use in early visual processing," *Proceedings of the Royal Society*, in press. Also Artificial Intelligence Laboratory, Memo No. 524, Massachusetts Institute of Technology, Cambridge, June 1979.

25. R. M. Mersereau, "The processing of hexagonally sampled two-dimensional signals," *Proc. IEEE*, vol. 67, pp. 930-949, June 1979.

26. W. E. Milne, *Numerical Solution of Differential Equations*, New York: Dover, 1979.

27. H.-H. Nagel, "Analyzing sequences of TV-frames," in *Proc. 5th Int. Joint Conf. Artificial Intelligence*, p. 626, August 1977.

28. H.-H. Nagel, "Analysis Techniques for Image Sequences," in *Proc. 4th Int. Joint Conf. Pattern Recognition*, Kyoto, Japan, November 1978.

29. K. Nakayama and J. M. Loomis, "Optical velocity patterns, velocity-sensitive neurons and space perception," *Perception*, vol. 3, pp. 63-80, 1974.

30. A. N. Netravali and J. D. Robbins, "Motion-compensated television coding, Part I," *Bell Syst. Tech. J.*, vol. 58, pp. 631-670, March 1979.

31. K. Prazdny, "Computing egomotion and surface slant from optical flow," Ph. D. Thesis, Computer Science Department, University of Essex, Colchester, England, 1979.

32. K. Prazdny, "Egomotion and relative depth map from optical flow," *Bio. Cybernetics*, vol. 36, pp. 87-102, 1980.

33. K. Prazdny, "The information in optical flows," Technical Report TR-915, Computer Vision Laboratory, University of Maryland, College Park, 1980.

34. R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems*. New York: John Wiley and Sons, Interscience, 1967.

35. D. L. Russell (ed.), *Calculus of Variations and Control Theory*. New York: Academic Press, 1976.

36. W. Yourgau and S. Mandelstam, *Variational Principles in Dynamics and Quantum Theory*. New York: Dover, 1979.

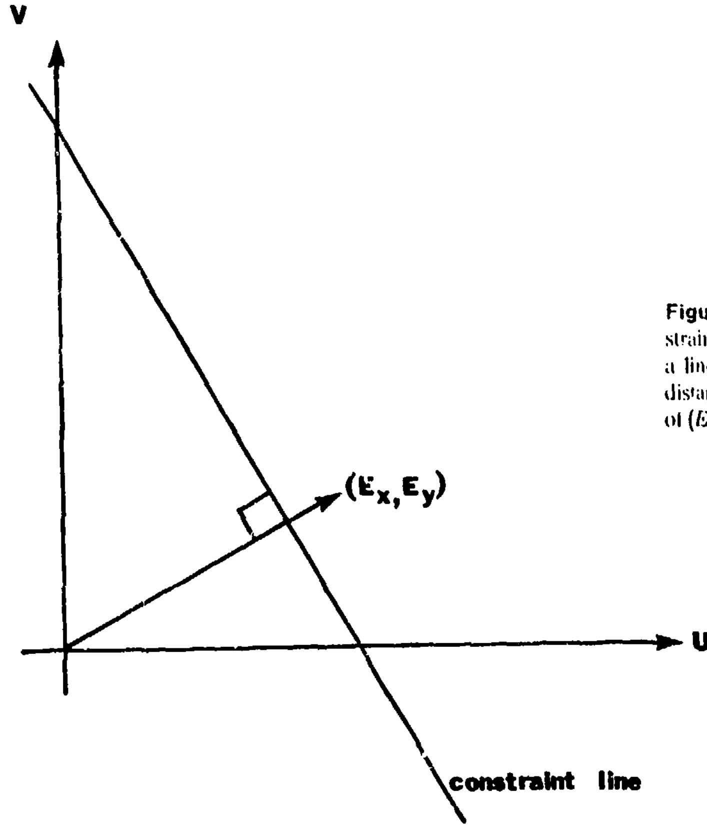
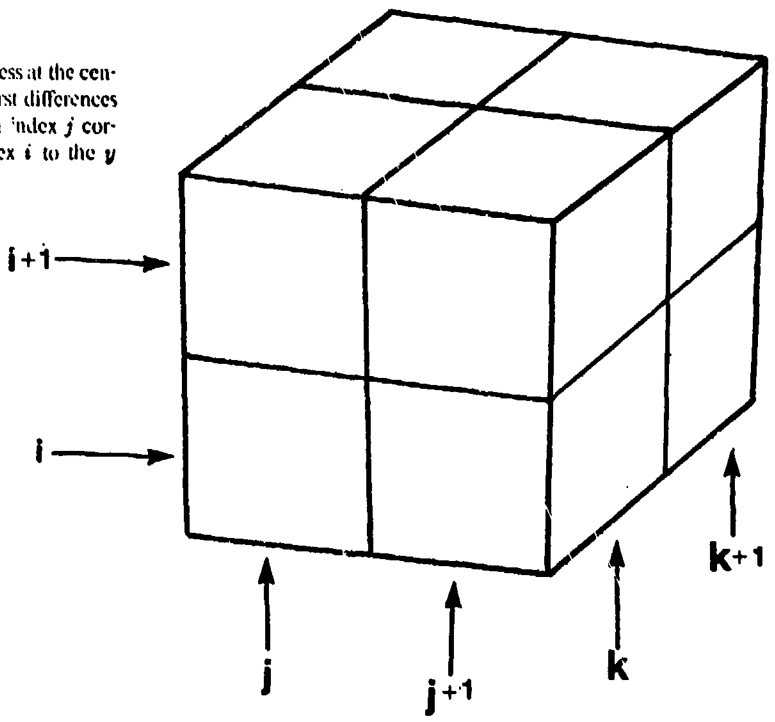


Figure 1. The basic rate of change of image brightness equation constrains the optical flow velocity. The velocity (u, v) has to lie along a line perpendicular to the brightness gradient vector (E_x, E_y). The distance of this line from the origin equals E_t divided by the magnitude of (E_x, E_y).

Figure 2. The three partial derivatives of image brightness at the center of the cube are each estimated from the average of first differences along four parallel edges of the cube. Here the column index j corresponds to the x direction in the image, the row index i to the y direction, while k lies in the time direction.



$^1/_{12}$	$^1/_6$	$^1/_{12}$
$^1/_6$	-1	$^1/_6$
$^1/_{12}$	$^1/_6$	$^1/_{12}$

Figure 3. The Laplacian is estimated by subtracting the value at a point from a weighted average of the values at neighboring points. Shown here are suitable weights by which values can be multiplied.

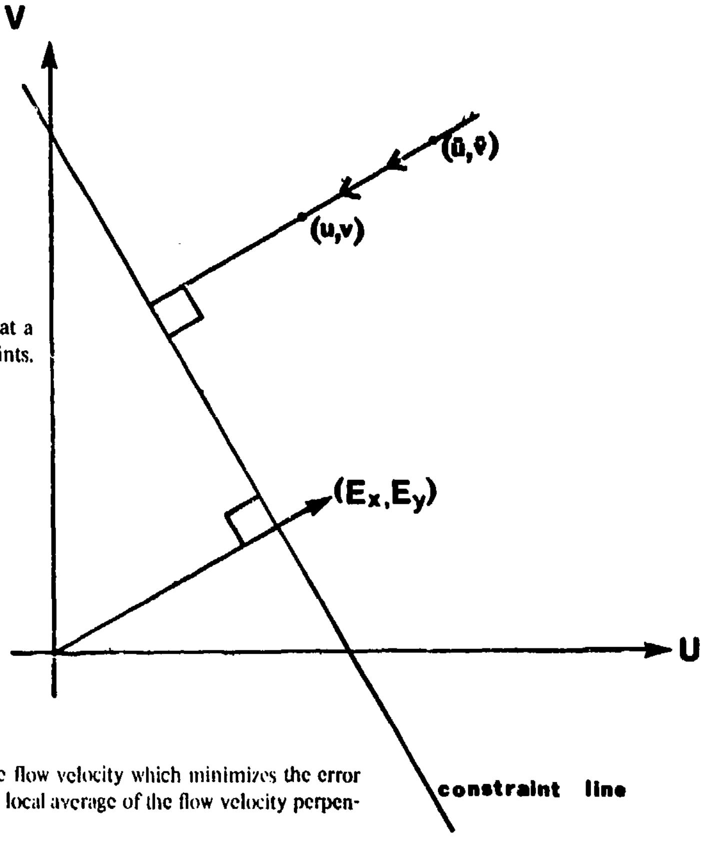


Figure 4. The value of the flow velocity which minimizes the error lies on a line drawn from the local average of the flow velocity perpendicular to the constraint line.

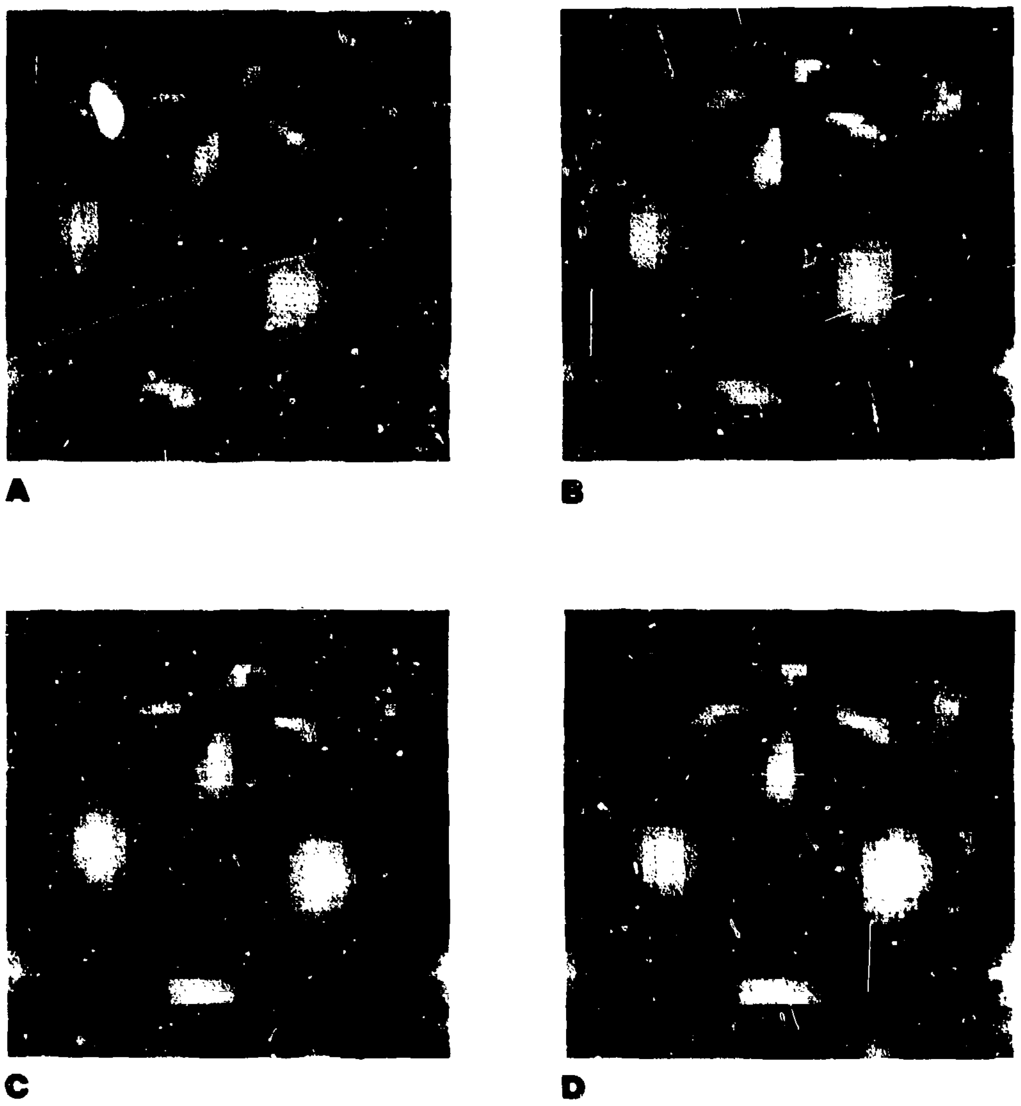


Figure 5. Four frames out of a sequence of images of a sphere rotating about an axis inclined towards the viewer. The sphere is covered with a pattern which varies smoothly from place to place. The sphere is portrayed against a fixed, lightly textured background. Image sequences like these are processed by the optical flow algorithm.

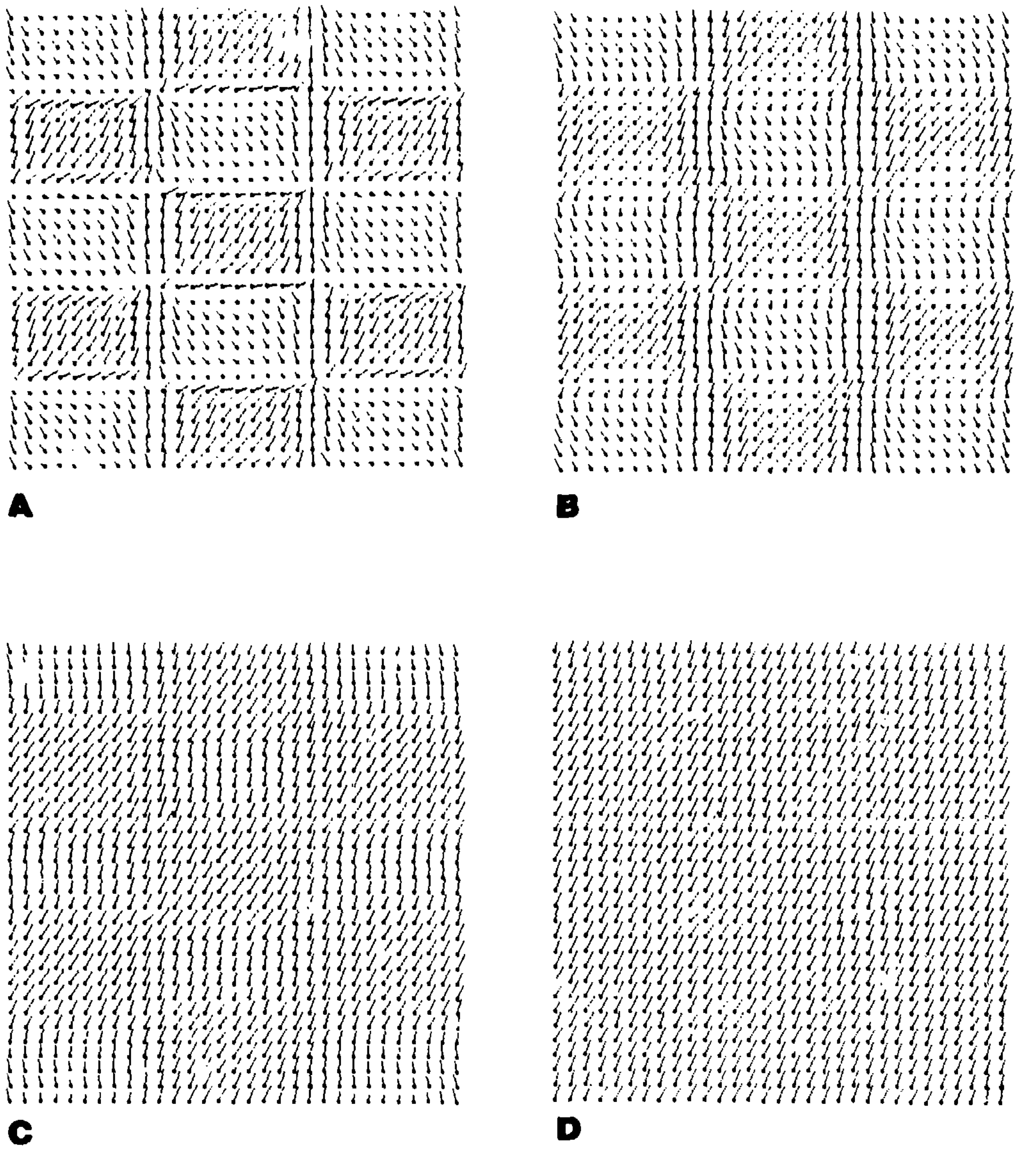
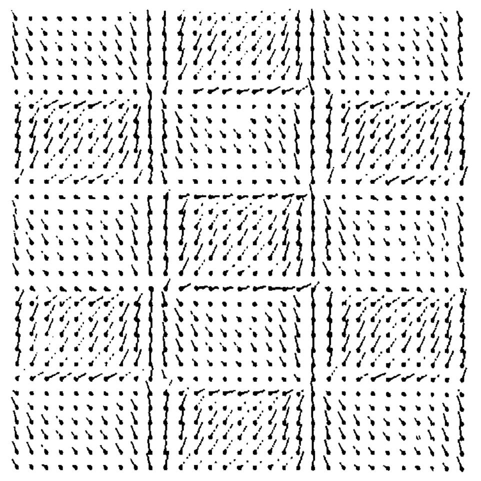
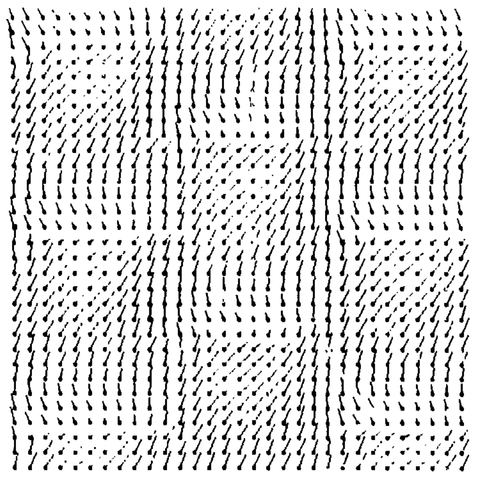


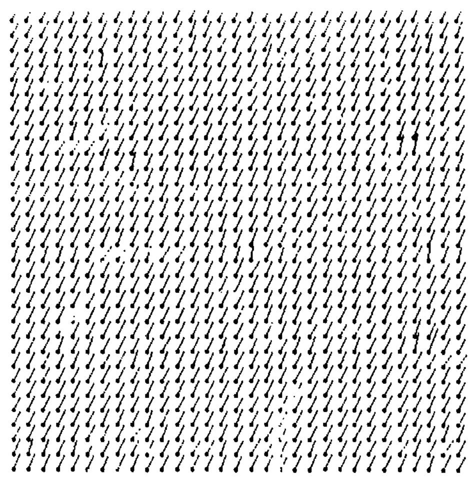
Figure 6. Flow pattern computed for simple translation of a brightness pattern. The estimates after 1, 4, 16, and 64 iterations are shown. The velocity is 0.5 picture cells in the x direction and 1.0 picture cells in the y direction per time interval. Two images are used as input, depicting the situation at two times separated by one time interval.



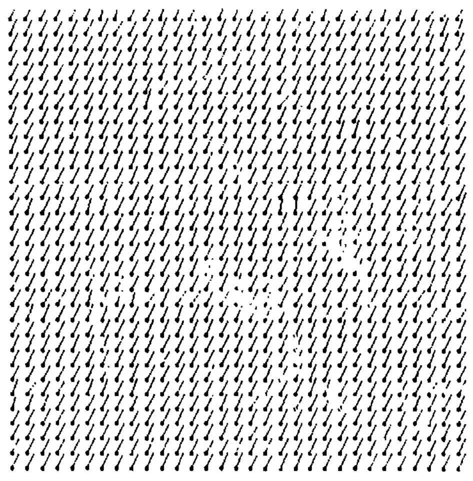
A



B



C



D

Figure 7. Flow pattern computed for simple translation of a brightness pattern. The estimates after 1, 4, 16, and 64 time steps are shown. Here one iteration is used per time step. Convergence is more rapid and the velocities are estimated more accurately.

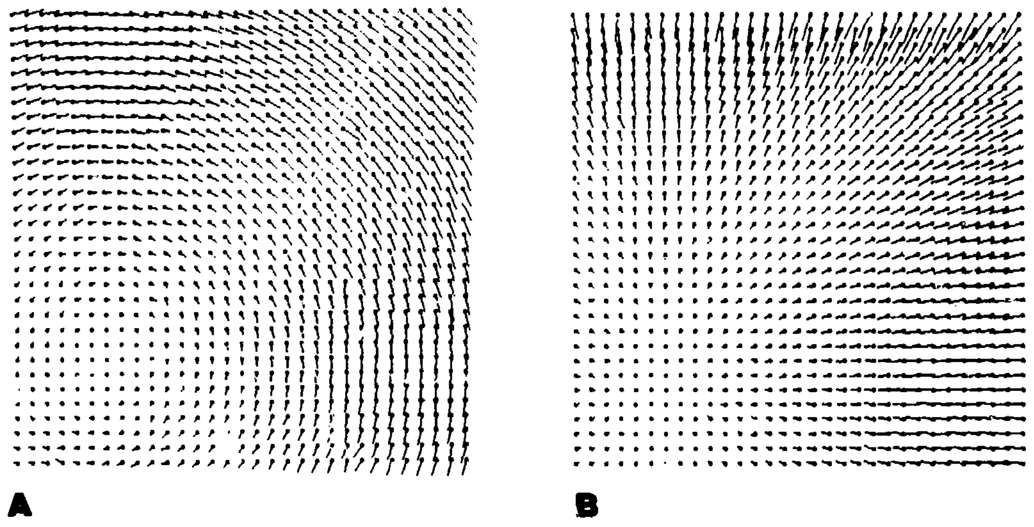


Figure 8. Flow patterns computed for simple rotation and simple contraction of a brightness pattern. In the first case, the pattern is rotated about 2.8 degrees per time step, while it is contracted about 5% per time step in the second case. The estimates aft[illegible] time steps are shown.

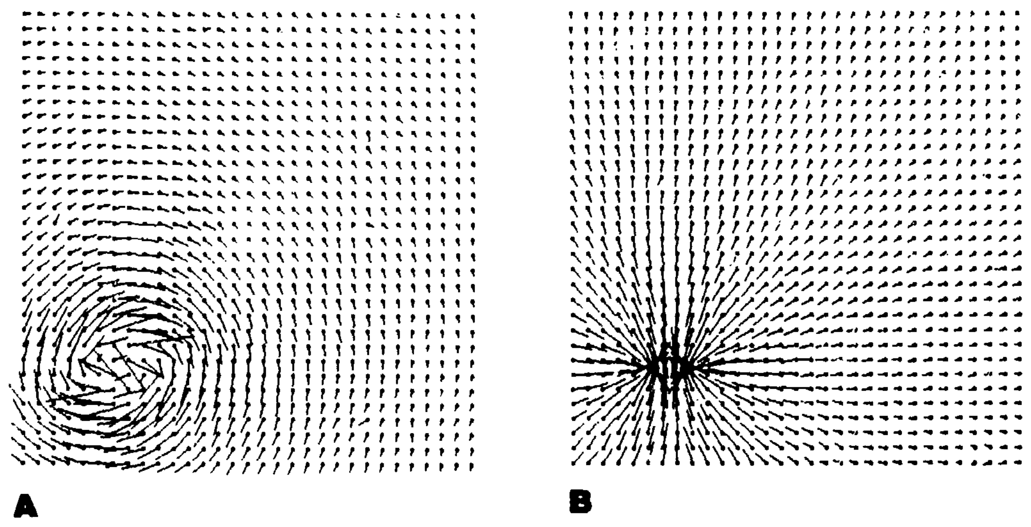
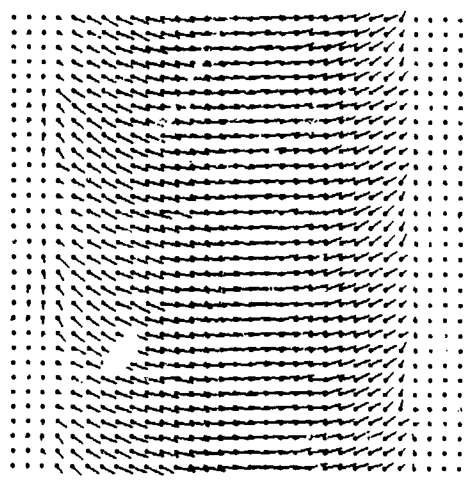


Figure 9. Flow patterns computed for flow around a line vortex and two dimensional flow into a sink. In each case the estimates after 32 iterations are shown.


A

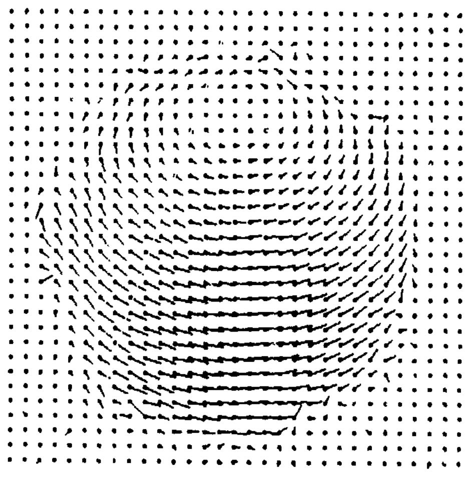
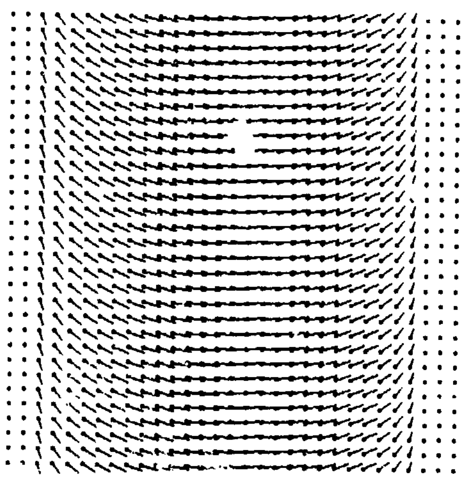

B

Figure 10. Flow patterns computed for a cylinder rotating about its axis and for a rotating sphere. The axis of the cylinder is inclined 30 degrees towards the viewer and that of the sphere 45 degrees. Both are rotating at about 5 degrees per time step. The estimates shown are obtained after 32 time steps.


A

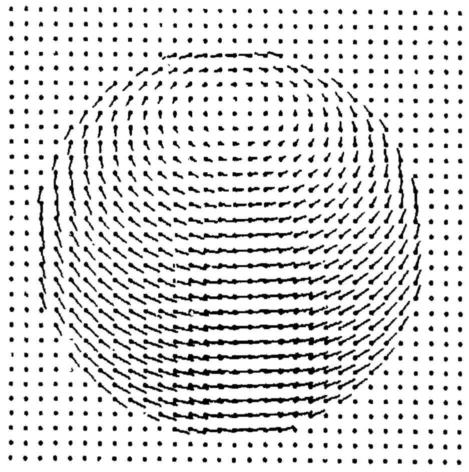

B

Figure 11. Exact flow patterns for the cylinder and the sphere.

COMPUTATIONAL STEREO FROM AN IU PERSPECTIVE*

Stephen T. Barnard
Martin A. Fischler

SRI International, Menlo Park, California

I INTRODUCTION

This paper surveys and evaluates computational methods for the recovery of depth information from multiple images. We identify the major functional components that comprise these methods, list various alternatives for implementing them, and analyze the domain-dependent and application-dependent constraints that favor some alternatives over others. Finally, we outline a program for evaluating the various components and systems on the IU Testbed.

The scope of this paper is restricted to research in the image understanding community. IU researchers have drawn on stereo work from other areas, especially cartography, psychology, and neurophysiology. We will not try to cover all the IU research relevant to stereo, but instead will select a cross-section of the most widely-known work that covers all the important and significantly different approaches to the stereo problem.

Much of the research in image-understanding has been devoted to recovering the range and orientation of surfaces and objects depicted in imaged data. The earliest work concentrated on an artificial domain -- the "blocks world." Significant (but not necessarily extendable) advances were made in this simple domain; in particular, it was shown that edge and vertex labeling schemes could provide constraints that allowed one to correctly partition a complex scene. More recent work, which has concentrated on real world problems, can be divided into three classes: (1) those that acquire range information directly with an active sensor, (2) those that depend on monocular information available in a single image (or perhaps several images from a single viewpoint under different lighting), and (3) those that use two or more images taken from different viewpoints and perhaps at different times. We are concerned here with this third class, which we shall refer to as "generalized stereo."

The generalized stereo paradigm includes conventional stereo, as well as what is often called motion parallax. In conventional stereo two images are recorded simultaneously, or nearly so, by laterally displaced cameras. In motion parallax two or more images are recorded sequentially, usually with a single camera that moves in an arbitrary direction. In a sense, conventional stereo can be considered to be a special case of motion parallax, and the same geometrical formalisms apply to both. In practice, they are often treated in different ways and are used in different domains; for example, motion parallax stereo forms the basis for most of the cartographic products derived from aerial surveys while conventional stereo is preferred for three-dimensional biological imaging systems.

Stereo is an attractive source of information for machine perception because it leads to direct range measurements, and, unlike monocular approaches, does not merely infer depth or orientation through the use of photometric and statistical assumptions. Once the stereo images are brought into point-to-point correspondence, the recovery of range values is a relatively straightforward matter. Furthermore, stereo is a passive method. Active ranging methods that use structured light, laser rangefinders, or other active sensing techniques are useful in tightly controlled domains, such as industrial automation applications, but are clearly unsuitable for most machine vision problems.

Perhaps the most common use of computational stereo is in the interpretation of aerial images. Other applications are passive navigation for autonomous vehicle guidance, industrial automation applications, and the interpretation of micro-stereophotographs for biomedical applications. Each domain has different requirements that can affect the design of a complete stereo system.

II THE COMPUTATIONAL STEREO PARADIGM

Research on computational solutions for the generalized stereo problem has followed a single paradigm, although there have been several distinct variations, both in method and intent. The paradigm involves the following steps.

* The Research described in this paper is based on work performed under Adavnced Research Projects Agency Contract No. MDA903-79-C-0588.

* Image acquisition

* Camera modeling

* Feature acquisition

* Matching

* Distance (depth) determination

* Interpolation

A. Image Acquisition

The most important factor affecting image acquisition is the specific application for which the stereo computation is intended. Three applications have received the most attention: the interpretation of aerial photographs for automated cartography, guidance and obstacle avoidance for autonomous vehicle control, and the modeling of human stereo vision.

Aerial photo-interpretation involves low-oblique and usually low-resolution images of a variety of terrain types. The stereo images are usually interpreted as pairs, rather than as longer sequences involving more than two images.

Stereo for autonomous vehicle control has been studied in two contexts: as a passive navigation aid for drone aircraft, and as a control system for surface vehicles. The images used for aircraft navigation are similar to the aerial photographs used for cartography, except that long sequences of images are used, and multispectral sensors are often employed. The images used for surface vehicle control are quite different -- they are high-oblique, comparatively high-resolution images.

Research on computational models of human stereo vision have mostly used synthetic random-dot stereograms as their subject matter for experimental investigation; the primary reason for this is that random-dot stereograms exclude all monocular depth cues, and the exact correspondences are known. Because the parameters of a random-dot stereograms, such as noise and density, can controlled, they allow systematic comparison of human and machine performance. Grimson has also reported experimental results with natural imagery [1].

Perhaps the most significant and widely recognized difference in scene domains is the difference between scenes containing cultural features such as buildings and roads, and those containing only natural objects and surfaces such as mountains, flat or "rolling" terrain, foliage, and water. Important stereo applications range over both domains. Low-resolution aerial imagery, for example, usually contains mostly natural features, although cultural features are sometimes found. Industrial applications, on the other hand, tend to involve man-made objects exclusively. Cultural features present special problems. For example, periodic structures such as the windows of buildings and road grids can confuse a stereo matcher. The relative abundance of occlusion edges in a city scene also causes problems because large portions of the images may be unmatchable. Cultural objects often have large surfaces with nearly uniform albedo that are difficult to match because of a lack of detail. Stereo systems that have been reported in the literature are usually targeted at a specific scene domain, and there is seldom any attempt to validate the methods in other domains.

In summary, the key parameters associated with image acquisition are:

* Scene domain

* Timing
  - Simultaneous
  - Nearly simultaneous
  - Radically different times

* Time of day (lighting and presence of shadows)

* Photometry (including spectral windows)

* Resolution

* Field of view

* Relative camera positioning (length and orientation, relative to the scene, of the stereo base line).

The issues associated with the scene domain are percentage of:

* Occlusion

* Man-made objects (straight edges, flat surfaces)

* Continuous surfaces of some minimal extent

* Textureless area

* Area containing repetitive structure.

B. Camera Modeling

Perspective geometry can be used to constrain the search for matches to one dimension. The extended line connecting the perspective centers of two cameras is called the air base; the points where the air base intersects the image planes are the epipoles; and a plane that contains the epipoles is an epipolar plane. Every point in one image of a stereo pair defines an epipolar plane, and the corresponding point in the other image must lie in the same plane. The search for a match of a point in the first image may therefore be limited to the line in the second image that is the intersection of the epipolar plane with the image plane, commonly called an epipolar line.

If the stereo pair is "perfect," the epipolar lines are coincident with the horizontal scan lines -- a convenient situation because the matching process can be made relatively simple and efficient. Stereo systems that have been primarily concerned with modeling human ability have used this approach [1,2]. In practical applications, however, the stereo pair may be imperfect. In aerial stereo photogrammetry, for example, the camera axis may typically be tilted as much as two to three degrees from vertical [3]. The implication here is that points on a scan line in one image will not fall on a single scan line in the second image of the stereo pair, and thus, the computational cost to employ the epipolar constraint is significantly increased.

The relative position of the two camera positions is called the camera model. Camera models are important because they allow one to exploit the epipolar constraint. In most cases, considerable a priori knowledge of the camera model is available, but it is often not as accurate as desired. Gennery [4] has developed a method for solving for the camera model from a relatively few sparse matches. His method accounts for differences in azimuth, elevation, pan, tilt, roll, and focal length.

Fischler and Bolles [5] have provided a number of results with respect to the minimum number of points needed to obtain a solution to the camera calibration problem, given a single image and a set of correspondences between points in the image and their spatial (geographic) locations; they also provide a technique for solving for the complete camera model, even when the given correspondences contain a large percentage of errors. While this work was directed at the problem of establishing a mapping between an image and an existing geographic database, it is obviously possible to apply the results to the stereo problem, and in fact, tying the stereo pair to an existing database offers the possibility of employing constraints beyond those available from the imaging geometry.

Camera modeling can be extended to include distortions introduced in the image-making process. Significant image distortion will degrade the accuracy of depth measurements made by a stereo system unless corrected. Two kinds of image distortion are found: radial and tangential. Radial distortion causes image points to be displaced perpendicular to the optical axis and may occur in the form of pin-cushion distortion (i.e., positive radial distortion) or barrel distortion (i.e., negative radial distortion). Tangential distortion is caused by imperfect centering of lens elements, resulting in image displacements perpendicular to the radial lines. Moravec described a method to correct for distortion using a square pattern of of dots [6]. Fourth degree polynomials are found that transform the measured positions of the dots to their nominal positions.

In summary, the important issues in camera modeling are:

* A priori knowledge of camera positions

* Solutions using a few sparse matches

* Use of known scene coordinates

* Ability to deal with matching errors

* Compensation for image distortion

C. <u>Feature Acquisition</u>

That featureless areas of nearly homogeneous brightness cannot be matched with confidence is widely recognized. Accordingly, most work in computational stereo has included some form of local feature detection, the particular form of which is closely coupled with the matching strategy used.

Approaches that apply area cross-correlation matching often use an interest operator to locate places in one image that can be matched with confidence. One way to do this is to select areas that have high variance. These areas will not be good features, however, if the variance is due only to brightness differences in the direction perpendicular to the epipolar line. These areas can be culled by demanding that the two-dimensional autocorrelation function have a distinct peak. Another widely used interest operator is the Moravec operator [6], which selects points that have high variance between adjacent pixels in four directions. Hannah has modified this operator to consider ratios of directional variances as well as ordinary isotropic variance over larger areas, and this modified operator seems to locate a better selection of both strong and subtle features.

Feature detection is more centrally important to those approaches that directly match features in the stereo images. The features that are used may vary in size, direction, and dimensionality.

Point-like features are good candidates for matching when the camera model is unknown and the matches are not constrained to epipolar lines. This is because, unlike linear features, points are unambiguously located in the image and can be matched in any direction. Linear features must be oriented across the epipolar lines if they are to be matched accurately. Another advantage of point-like features is that they can be matched without concern for perspective distortion. In area-correlation approaches point-like features are often used to obtain the camera model prior to more extensive matching. The local intensity values around a point can be used to establish initial confidences of matches in a way similar to area correlation [7].

If the camera model is known a priori or derived in a preliminary step, edge elements can be used as primitive matching features. The computational model of human stereo vision described by Marr and Poggio uses zero-crossings in the convolution of a circularly symmetric Laplacian with a Gaussian low-pass filtered image [1], [2]. The zero-crossings are found in the convolution of four differently sized masks. Arnold uses the Hueckel operator to find linear features, but the operator has a fixed size [9]. Baker uses zero-crossings of a one-dimensional operator, again of a fixed size [10].

Many distinct edge models have been proposed as the basis for edge-detecting algorithms. In the case of "strong" edges, most of the resulting algorithms yield similar results for operators of comparable sizes. Often the same underlying model appears in different implementations; e.g., zero-crossings in the second derivative are equivalent to local maxima in the first derivative, and most of the conventional edge detection methods search for approximations to first derivative maxima. More important is how the edge attributes can be used for matching; size, direction, and magnitude (contrast) have been used, but their relative merit is not established.

For the most part, low level features have been used for stereo. What we mean by "low level" is that the features depend only on local monocular intensity patterns, and are based on the assumption that more-or-less sharp intensity gradients are usually due to physically significant structural, reflectance, and illumination events in the scene. Higher level features that depend on more sophisticated semantic analysis have been largely unused (Ganaparthy described a system for matching vertices in blocks-world stereo scenes across very large viewing angles [11]). The ability to classify edges as occlusion or nonocclusion boundaries, for example, could be very useful to a stereo system, especially in the difficult domains that include a wealth of cultural features.

In summary, the properties of local features that are important to the computational stereo problem are:

* Dimensionality (point-like versus edge-like)

* Size (spatial frequency)

* Magnitude (contrast)

* Semantic content

* Density of occurence

* Easily measurable attributes

* Uniqueness/distinguishability.

D. Matching

Image matching is a core area in scene analysis and will not be covered in in full detail in this paper. Instead, we will focus on those portions of the image-matching problem that are directly relevant to stereo modeling. Features that distinguish stereo image matching from image matching in general are the following:

* Images are taken at approximately the same time and from the same viewpoint in space. Thus, illumination/shadow conditions are the same (although there can be significant differences in specular reflection). Most of the significant changes will occur in the appearance of nearby objects and in occlusions. Additional changes in both geometry and photometry can be introduced in the film development and scanning steps, but can usually be avoided by careful processing.

* Stereo modeling generally requires that a dense grid of points be matched.

Ideally, we would like to find the correspondences (i.e., the matched locations) of every individual pixel in both images of a stereo pair. However, it is obvious that the information content in the intensity value of a single pixel is too low for unambiguous matching. In practice, coherent collections of pixels are matched. These collections are determined and matched in two distinct ways (see the discussion in the preceding section on feature acquisition):

* Area Matching: Regularly sized neighborhoods of a pixel are the basic units that are matched. This approach is justified by the "continuity assumption," which asserts that at the level of resolution at which stereo matching is feasible, most of the image depicts portions of continuous surfaces; therefore, adjacent pixels in an image will generally represent contiguous points in space. This approach is almost invariably accompanied by correlation matching to establish the correspondences.

* Feature Matching: "Semantic features" (with known physical properties and/or spatial geometry), or "intensity anomaly features" (isolated anomalous intensity patterns not necessarily having any physical significance), are the basic units that are matched. Semantic features of the generic type include occlusion edges, vertices of linear structures, and prominent surface markings; domain-specific semantic features might include such features as the corner or peak of a building, or a road surface marking; intensity anomaly features include those such as zero crossings and image patches found by the Moravec interest operator. Methods used for feature matching often include symbolic classification techniques, as well as correlation.

Obviously, feature matching alone cannot provide the desired dense depth map so it must be augmented by a model-based interpretation step (e.g., we recognize the edges of buildings and assume that the intermediate space is occupied by planer walls and roofs), or by area matching. When used in conjunction with area matching, the feature matches are generally considered to be more reliable and can constrain the search for correlation matches.

To further reduce the possibility of error caused by an ambiguous match, a number of hierarchical and global matching techniques have been employed, including relaxation matching and various "coarse-fine" hierarchical matching strategies.

The correlation-matching approach attempts to resolve ambiguity by using as much local information as possible to make decisions about potential matches, but each match decision is made independently of the others. The relaxation-labeling approach uses a relatively small amount of local information for each potential match, and attempts to resolve ambiguity by finding consensuses among subsets of the total population of matches, relying on the three-dimensional continuity of surfaces to be reflected in the two-dimensional continuity of disparity. A method for avoiding ambiguity that can be applied to both correlation matching [6] and feature point matching [2] is the so-called "coarse-fine" strategy. In this approach coarse disparities are found relatively quickly, but with low accuracy. These gross disparities are used to bias finer-resolution matching. Even with a coarse-fine strategy, however, some ambiguity at each level of resolution is inevitable. The best combination of ambiguity avoidance and ambiguity resolution is a major research issue.

In summary, key attributes which differentiate matching techniques include:

* Local versus global ambiguity resolution
* Area (dense) versus feature (sparse) matching.

The constraints used to both limit computation and reduce ambiguity include:

* Epipolar
* Continuity
* Hierarchical (e.g., coarse-fine matching)
* Sequential (e.g., feature tracking in sequential views).

Criteria that can be used to evaluate (or compare) different matching techniques include:

* Accuracy (match precision to the sub-pixel level)
* Reliability (resistance to gross classification errors)
* Generality (applicability to different scene domains)
* Predictability (availability of performance models)
* Complexity (cost of implementation; computational requirements).

E. Distance Determination

With few exceptions, work in image understanding has not dealt with the specific problem of distanca determination. The matching problem is has been considered the hardest and most significant problem in computational stereo. Once accurate matches have been found the determination of distance is a relatively simple matter of triangulation; nevertheless, this step presents significant difficulties, especially if the matches are somewhat inaccurate or unreliable.

To a first approximation, the accuracy of stereo distance measurements is directly proportional to the accuracy of the matches and inversely proportional to the length of the stereo baseline. We have discussed how lengthening the stereo baseline complicates the matching problem by increasing ambiguity, and how various matching strategies have been used to overcome this problem (coarse/fine strategies, cooperative or relaxation-labeling approaches, and incremental stereo views). The role that accuracy of matches plays has been less thoroughly examined.

In many cases, matches are made to an accuracy of only a pixel. However, both the area correlation and the feature-matching approaches can lead to better accuracy. Sub-pixel accuracy using area correlation requires expensive interpolation over correlation patches, however, and also complicates feature-matching approaches.

Another approach is to settle for one-pixel accuracy, but to use multiple views [6]. A match from a particular pair of views represents a depth estimate with uncertainty that depends on the one-pixel accuracy of the match and on the length of the stereo baseline. Matches from many pairs of views can be statistically averaged to find a more accurate estimate. The contribution of a match to the final depth estimate can be weighted according to any factors that bear on the confidence of the match and on its accuracy.

In summary, better depth measurements can be obtained in several ways, each involving overhead:

* Sub-pixel matching
* Increased stereo baseline
* Statistical averaging over several views.

F. Interpolation

As previously mentioned, stereo applications usually demand a dense array of depth estimates that the feature matching approach cannot provide because features are sparsely and irregularly distributed over the images. The area correlation-matching approach is more suited to obtaining dense matches, although it tends to be unreliable in areas of low information. Consequently, some kind of interpolation step is required.

The most straightforward way to create the dense depth array from a sparse array is simply to treat the sparse array as a sampling of a continuous depth function, and to approximate the continuous function using a conventional interpolation method (for example, by fitting splines). Assuming the sparse depth array is complete enough to capture the important changes in depth, satisfying the conditions of the sampling theorem, this approach may be adequate. Aerial stereophotographs of rolling terrain, for example, might be handled in this way. In many applications, however, the continuous depth function model will not be appropriate because of occlusion edges.

A more sophisticated approach to the interpolation problem is to fit a priori geometric models to the sparse depth array. Normally, model fitting would be preceded by clustering to find the subsets of points that correspond to significant structures. Each cluster would then be fit to the best available model, thereby instantiating the model's free variables and providing an interpolation function. This approach has been used to find ground planes [9], elliptical structures in stereophotographs [12], and smooth surfaces in range data acquired with a laser rangefinder [13].

III EVALUATION

The following criteria are appropriate for evaluating both complete stereo systems and the components of such systems. More specialized criteria relevant to individual components of stereo systems were presented in previous sections of this paper.

(1) Disparity - what range of disparity is handled? One possible advantage of automated stereo analysis is that computer methods may be able to handle larger angular disparities than humans can. Larger disparities lead to more accurate depth measurements, but also to more difficult matching.

(2) Coverage - what precentage of the scene is matched? Also, how widely are the matches distributed? Clearly, large, featureless, homogeneous areas cannot be readily matched. What kinds of interpolation techniques can be used to extend disparity across such areas? What monocular techniques can be used to enhance coverage (for example, photometric evidence for smooth surfaces)?

(3) Accuracy

(4) Reliability - how many false matches are made compared to valid matches? What methods are effective for detecting and eliminating false matches?

(5) Domain sensitivity - what range of scene domains can be handled?

(6) Efficiency - actual timings of stereo systems will probably not be useful because of nonoptimal implementations and differences in hardware. Comparisons based on computational complexity can be made, however. How does the time required for stereo compilation scale with the image size, with the range of disparity, and with other important parameters? How amenable to hardware implementation are the different methods? What efficiency is needed for useful automated stereo systems?

(7) Human engineering - how are the results displayed (perspective 3D plots, false coloring, countour plots, vector fields, etc.)? What are the best methods? Is human interaction allowed?

(8) Sources of data for experimental validation

(a) Synthetic images or scaled models (model boards)
- * Advantages:
  - Cheap
  - Certainty about actual depths
  - Control over secondary parameters
- * Disadvantages:
  - Not as complex as real-world scenes
  - Not representative of any real image domain

(b) Ground surveys.
- * Advantages:
  - Realistic
  - Certainty about actual depths
- * Disadvantages:
  - Expensive (hence limited number of sites that can be surveyed)

(c) Compare to human performance.
- * Advantages:
  - Realistic
  - Reasonably inexpensive
- * Disadvantages:
  - Susceptible to human errors

IV SURVEY

This survey covers a representive sampling of the image understanding work relevant to computational stereo. While not exhaustively covering the field, it does contain examples of all the significantly different approaches to the steps in the computational stereo paradigm. The work dascussed in the survey is roughly grouped according to the research centers where the primary investigators are resident, although exceptions will be found. Other organizations were considered, but none was entirely satisfactory.

Control Data Corporation

A flexible approach to digital stereo mapping, [14]

This work is concerned with the automation of stereo-mapping functions. The primary concerns have been with handling different kinds of terrain and sensors, efficient hardware implementation, and the develonment of an interactive mapping system.

A regularly spaced grid of points in the left image is matched in the right image. Matching is done by searching along the corresponding epipolar line in the right image for a maximum correlation patch, which is warped to account for predicted terrain relief (estimated from previous matches). Sub-pixel matches are obtained by fitting a quadratic to the correlation coefficients and picking the interpolated maximum.

"Tuning parameters" may be dynamically altered to adapt the system to sensor and terrain variations. Some tuning parameters are grid limits and interval sizes; patch size and shape; number of correlation sites along the search segment; prediction function weighting coefficients; and reliability thresholds for the correlation coefficient, standard deviation, prediction function range, and slope of the correlation function. The intent is to choose the smallest feasible patch, subject to the need to compensate for noise and lack of intensity variation in the image.

A continuity constraint is used to predict matches. The rate of change of disparity is assumed to be continuous. This constraint is also used to shape the correlation patches in the left image (using a linear interpolation and bi-linear resampling).

The reliability of matching is continuously monitored to signal when parameters become inappropriate or when the photometry prevents valid matching. Reliability is estimated with a combination of correlation coefficient, patch standard deviation (are features present?), distance of maximum from predicted point, prediction function limits, and slope of the correlation function.

The system is implemented on a highly parallel configuaration of 4 CDC Flexible Processors, each capable of 8 MIPS.

A somewhat different approach has been taken for three-dimensional modeling of cultural sites (e.g., building complexes) from high-resolution images. The basic idea is to identify corresponding points of intersection between epipolar lines and edges in the two images of a stereo pair. Non-matched edges are assumed to be due to noise or occlusions. Depth along an epipolar line (corresponding to a three-dimensional profile line in the scene) is assumed to vary linearly between contiguous pairs of matched intersections. Special techniques are developed to deal with occlusions and "reversals." Edge-tracking across sequential epipolar lines (the continuity constraint) contributes to reliability.

Lockheed

Bootstrap stereo, [15]

The goal of this study is navigation of an autonomous aerial vehicle using passively sensed images, using a method called "bootstrap stereo." Ground control points are used to determine the vehicle's location and a camera model is used to locate further correspondences. Major components of the system are camera calibration, new landmark selection, matching, and control point positioning. The complete system will consist of several navigation "specialists," including ones using instrumentation (altimeter, airspeed indicator, attitude gyros), dead reckoning, landmarks, and stereo.

Camera calibration is achieved with standard least-squares methods to determine position and orientation of the camera.

New landmark selection involves an adaptation of the Moravec operator that uses ratios of directed variance along two orthogonal directions (instead of simply the directed variance in four directions).

Point matching is accomplished with normalized cross-correlation using a spiraling grid search. Coarse matching is used to approximately register the images and to initialize second-order prediction polynomials. Autocorrelation thresholding is used to evaluate the reliability of matches (Good matches have sharply peaked autocorrelation functions.). Subpixel matching accuracy is achieved through parabolic interpolation of the correlation values.

Control point positioning involves determining the depth of matched points. It is done with straightforward triangulation.

Stanford

(1) Stereo-camera calibration, [4]

A method for determining the relative position and orientation of two cameras from a set of matching points is developed. The calibration accounts for difference in azimuth, elevation, pan, tilt, roll, and focal length. The basic method is a least-squares minimization of the errors of the distances of points in image 2 from their predicted locations, as determined by their positions in image 1 and an estimated camera model. The nonlinear optimization problem is solved by iterating on a linearization of the problem.

(2) Local context in matching edges for stereo vision, [9]

This approach matches corresponding features instead of matching areas using cross correlation. Features are edge elements produced by a Hueckel operator. The approach uses a continuity constraint to resolve ambiguity. If a scene is continouous in three dimensions then adjacent matching edge elements should be continuous in direction and disparity. Intensities on either side of the edge should also be consistent.

The Moravec operator is used to select about 50 points. A coarse/fine search finds matches for some of these points, and Gennery's camera model solver is used to determine the parameters that relate the two camera positions. A ground plane is fit to the matches (few points may lie below the ground plane, some may be above it, and as many as possible lie on it). A Hueckel operator is applied to both images (3.19 pixel radius), and the results are transformed into a normalized coordinate system such that the stereo axis is in the x direction and points on the ground plane have zero disparity.

Each edge element in the left picture is matched to nearby candidates in the right image (there are usually about eight candidates) based on the angle and brightness information supplied by the Hueckel operator. Each edge in the left image is then linked to all its neighbors that seem to arise from the same physical edge. (Two edges are neighbors if they are close, have roughly the same angle, and similar brightness. Three or four are typically found.) The linked neighbors of an edge element vote to determine which of the candidate disparities is most consistent.

Some problems caused by the Hueckel operator are identified (for example, it is unreliable for corners, textured areas, and slow gradients). The author suggests relaxation as a way to use context in a more controlled way (see [7]). The system works well in scenes of man-made objects, but poorly in natural scenes (the opposite of area correlation).

(3) Object detection and measurement using stereo vision, [12]

This study uses stereo or rangefinder data to detect and measure objects, and although it does not deal with the matching problem, it is relevant to the interpolation and interpretation problems. The system is intended for autonomous vehicle guidance and obstacle avoidance.

First, the ground surface is found as described by Arnold in [9]. Above-ground points are clustered and ellipsoids are fit. Clustering is done with a minimal spanning tree approach. The author suggests the use of relaxation for clustering. Next, the ellipsoids are adjusted to a better fit with a modified least-squares method. Two types of errors are considered: the amount by which the points in a cluster being fit miss lying on the ellipsoid, and the amount by which the ellipsoid occludes any points as seen from the camera. (Orthographic projection, not central projection, is assumed.) In addition, there is an a priori bias to make any small ellipsoids approximately spherical.

After ellipsoids have been fit to the original clusters, it may become apparent that the initial clustering, based on only local information, did not produce a good segmentation. In this case, the initial clusters are either split or merged and another set of ellipsoids is fit to them.

Although this work does not address the central problems of computational stereo, it is an interesting way of both smoothing and interpreting raw depth information made available from stereo. The ellipsoid model is plausible for moon rocks, but probably not for most other objects.

(4) Visual mapping by a robot rover, [6]

This is a study of autonomous vehicle guidance. Severe noise problems are overcome by use of redundancy. An early approach that used only motion stereo was found to be unworkable because of matching errors and uncertain camera models. The latest approach uses "slider stereo" to obtain nine stereo views at 6.4 cm intervals. A calibration step determines the camera's focal length and distortion from a digitized test pattern.

An interest operator is used to select good features for matching. First, for each point it computes the variance between adjacent pixels in four directions over a square (3x3 pixel) neighborhood; next, it selects the minimum variance as its interest measure; and finally, it chooses feature points where the interest measure is locally maximal. Intuitively, each chosen point must have relatively high variance in several directions, and must be more "interesting" than its immediate neighbors. The interest operator is used on reduced versions of the images.

A binary search correlator matches 6x6 areas denoted by features in one image to areas in another image. The search begins at the lowest resolution (x16 reduction) and proceeds to the higher resolutions. In this way, points chosen from the center view are found in the other eight views. The uncertainty of the depth measurement associated with a match is inversely proportional to the length of the stereo baseline. To obtain more accurate depths, the measurements are averaged by considering each of the stereo baselines obtained from the thirty-six combinations of nine views taken two at a time. A measurement from a particular pair contributes a normal distribution with a mean at the estimated distance and a standard deviation inversely proportional to the stereo baseline. The curves are also normalized according to the correlation coefficients of the matches (a low coefficient reduces the area) and according to the degree of y-disparity (a large y-disparity reduces the area). The peak in the sum of these distributions gives a very reliable depth measurement.

Depth measurements are used to drive the vehicle in about one-meter increments. Vehicle motion is deduced from depth measurements at two positions by comparing the differences of point positions, which should be the same in both views. Each point is modeled as a sphere whose size depends on the uncertainty of the point's position. The vehicle is modeled as a three-meter sphere. From these, a near optimal path is determined to a goal point.

MIT

(1) Cooperative computation of stereo disparity, [16]

A parallel, "cooperative" computation model for human stereo vision is proposed. This feature matching method uses two constraints to match random-dot stereograms. The features that are matched are the dots themselves. The constraints are: uniqueness, which requires that every feature have a unique disparity (a consequence of imaged points on three dimensional surfaces having unique depths); and continuity, which requires that disparity varies smoothly almost everywhere (except at relatively rare occlusion boundaries). These constraints are applied locally over several iterations with an algorithm very much like relaxation-labeling. Multiple disparity assignments of points inhibit one-another, and local collections of similar disparities support one-another. Although this algorithm successfully fused random-dot stereograms, the authors rejected it as a model of human stereopsis and proposed a new model described below.

(2) A computational theory of human stereo vision, [2]
A computer implementation of a theory of human stereo vision, [17]
Aspects of a theory of human stereo vision, [1]

Matching of features occurs in four independent channels tuned to different spatial frequencies. The matches found in the lower frequency channels establish a rough correspondence for the higher frequency channels, thereby reducing the number of false matches.

In the original theory, the features that were proposed were zero-crossings of an image first low-pass filtered and then convolved with bar masks of four different sizes and different orientations, with a cross section that was a difference of Gaussian functions with space constants in the ratio of 1:1.75. The zero-crossings after a second difference operation correspond to extrema after a first difference operation. This method is therefore a way of finding edges at different scales. In the implementation of the theory bar masks were not used; instead, circularly symmetric differences of Gaussians were used to approximate the Laplacian of a Gaussian distribution. The convolutions were done on a LISP machine and special-purpose hardware. In the original theory line terminations were to be used as features, along with zero-crossings, but this has not been implemented.

Zero-crossings where the gradient is oriented vertically are ignored (The implicit camera model has the epipolar rays oriented horizontally.). Other zero-crossings are located to an accuracy of one pixel and their orientations (determined by the gradient of the convoluton values) is recorded in increments of 30 degrees.

Matching within any given channel proceeds independently of other channels. First, the "eye position" is fixed and a zero-crossing is located in one image. (The eye position is effectively a rigid translation of the two images with respect to one another, and defines a continuous mapping of points in one image to points in the other.) The region surrounding the corresponding point in the second image is then divided into three pools -- two larger convergent and divergent regions (towards and away from the "nose", respectively) and a smaller null-vergence region centered on the predicted match location. The pools together span a region of twice the width of the central positive region of the convolution mask. Zero-crossing from pools in the second image can match the one from the first image only if they result from convolutions of the same size mask, have the same sign, and have approximately the same orientation. If a unique match is found (i.e., only one of the pools has a zero-crossing satisfying the above criteria), the match is accepted as valid. If two or three candidate matches are found, they are saved for future disambiguation. Once all matches have been found (ambiguous or not), the ambiguous ones are resolved by searching through the neighborhoods of points to determine the dominant disparity (convergent, divergent, or null). This is the familiar continuity constraint.

It may be the case that the disparity of a region is greater than the range handled by the matcher. This is detected from the percentage of unmatched zero-crossings. Marr and Poggio showed that the probability of a zero-crossing having at least one candidate match in this situation is about 0.7. If the disparity is within the range of the matcher, however, the probability is much higher.

The lower frequency matching channels are used to guide the "eyes" to bring the higher frequency channels into range. The possibility of using other sources of information to guide the eye movement (in particular, texture contours) was mentioned by Grimson [1].

(1) Parametric correspondence and chamfer matching: two new techniques for image matching, [18]

A method for matching images to a three-dimensional symbolic reference map is presented. The reference map includes point landmarks, represented with three-dimensional coordinates; linear landmarks, represented as curve fragments with lists of three-dimensional coordinates; and volumetric structures, represented as wire-frame models. A predicted image is generated from an expected viewpoint by projecting three-dimensional coordinates onto image coordinates, suppressing hidden lines. The predicted image is matched to image features, and the error is used to adjust the viewpoint approximation. The matching is done by "chamfering." The image feature array is first transformed into an array of numbers representing the distance to the nearest feature point. The similarity measure is then computed by summing the distance array values at the predicted feature locations.

(2) The SRI road expert: image-to-database correspondence, [19]

The problem of matching an image to a database is studied. The images may be vary for several reasons: different camera parameters, lighting conditions, cloud cover, etc. The method that is presented begins with an estimate of the camera parameters, including estimates of uncertainties. It refines the estimated correspondence by locating landmarks in the image and comparing their image locations to their predicted locations. The uncertainties of the camera parameter estimates are modeled as a joint normal distribution. This model implies elliptical uncertainty regions in the image. The location of one feature constrains the uncertainty of others to relative uncertainty regions (These are also ellipses, but are usually significantly smaller than the unconstrained regions). Two kinds of matches between landmarks and image features are used: point-to-point and point-on-a-line. The point-to-point matches yield more information for refining the camera parameters, but the point-on-a-line matches are more numerous and cheaper to find. A modified version of Gennery's calibration method [4] is used to refine the camera parameters.

(3) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, [5]

A method for fitting a model to experimental data is developed (RANSAC) and applied to the "location determination problem" (i.e., given a set of control points with known positions in some coordinate frame, determine the locations from which an image of the control points was obtained). The method is radically different from conventional methods, such as least squares, which begin with large amounts of data and then attempt to eliminate invalid points. RANSAC uses a small, randomly chosen set of points and then enlarges this set with consistent data when possible. This strategy avoids a common problem with least squares and similar methods -- a few gross errors, or even a single one, can lead to very bad solutions. In practice, RANSAC can be used as a method for selecting and verifying a set of points that can be confidently fit to a model with a conventional method.

(4) Disparity analysis of images, [7]
The Image Correspondence Problem, [8]

Points are matched in two images that differ because of normal stereo, camera motion, or object motion. The Moravec operator is used to select point features in both images. An initial collection of possible matches is established by linking each point in the first image with possible matching points in the second image. (A point in the second image is considered a possible match if it is in a square area centered on the position of the point in the first image.) Each point from the first image is considered an object that is to be classified according to its disparity, and each of its possible matches establishes a label denoting one of several possible classifications. Each object also has a special label denoting "no-match." An initial confidence for each disparity label is determined based on the mean-square-difference of small regions surrounding the possible matching points. The estimates are iteratively improved with a relaxation-labeling algorithm that uses the continuity constraint. Support for each label of a particular object is calculated from the neighboring objects. If relatively many nearby objects have similar labels with high confidence, the label is strongly supported and its confidence increases. If no labels are strongly supported, the confidence of the "no-match" label increases. After a few iterations (about 8) the confidence estimates converge to unique disparity classifications for each point. (Convergence is not guaranteed theoretically, but is observed experimentally).

REFERENCES

1. W. E. L. Grimson, "Aspects of a computational theory of human stereo vision," Proc: Image Understanding Workshop, April 1980, pp. 128-149.

2. D. Marr and T. Poggio, "A theory of human stereo vision," Artificial Intelligence Lab., M.I.T., Cambridge, Massachusetts, Memo 451, November 1977.

3. Manual of Photogrammetry, third edition, M.M. Thompson, ed., American Society of Photogrammetry, Falls Church, Virginia, 1966.

4. D. Gennery, "Stereo-camera calibration," Proc.: Image Understanding Workshop, November 1979, pp. 101-107.

5. M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Technical Note 213, Artificial Intelligence Center, SRI International, Menlo Park, California, March 1980.

6. H. Moravec, "Visual mapping by a robot rover," Proc. 6th Int. Joint Conf. Artificial Intell., vol. 1, Tokyo, Japan, August 1979, pp. 598-600.

7. S. T. Barnard and W. B. Thompson, "Disparity analysis of images," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, no. 4, July 1980, pp. 333-340.

8. S. T. Barnard, The Image Correspondence Problem, Ph.D. dissertation, University of Minnesota, Minneapolis, Minnesota, December 1979.

9. D. Arnold, "Local context in matching edges for stereo vision," in Proc.: Image Understanding Workshop, May 1978, pp. 65-72.

10. H. Baker, "Edge-based stereo correlation," Proc: Image Understanding Workshop, April 1980, pp. 168-175.

11. S. Ganaparthy, "Reconstruction of scenes containing polyhedra from stereo pairs of views," Artificial Intelligence Lab., Stanford University, Stanford, California, Memo AIM-272, December 1975.

12. D. B. Gennery, "Object detection and measurement using stereo vision," Proc: Image Understanding Workshop, April 80, pp. 161-167.

13. R. O. Duda, D. Nitzan, and P. Barrett, "Use of range and reflectance data to find planar surface regions," Technical Note 162, Artificial Intelligence Center, SRI International, Menlo Park, California, April 1978.

14. R. L. Henderson, W. J. Miller, and C. B. Grosch, "A flexible approach to digital stereo mapping," Photogrammetric Engineering and Remote Sensing, vol. 44, no. 12, December 1978, pp. 1499-1512.

15. M. J. Hannah, "Bootstrap stereo," Proc: Image Understanding Workshop, April 1980, pp. 201-208.

16. D. Marr and T. Poggio, "Cooperative computation of stereo disparity," Science, vol. 194, pp. 283-287, 1976.

17. W. E. L. Grimson and D. Marr, "A computer implementation of a theory of human stereo vision," Proc.: Image Understand [illegible] orkshop, April 1979, pp. 41-47.

18. H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: two new techniques for image matching," Technical Note 153, Artificial Intelligence Center, SRI International, Menlo Park, California.

19. R.C. Bolles, L.H. Quam, M.A. Fischler and H.C. Wolf, "The SRI road expert: image-to-database correspondence," Proc.: Image Understanding Workshop, November 1978, pp. 163-174.

GEOMETRIC CONSTRAINTS FOR INTERPRETING IMAGES OF COMMON STRUCTURAL ELEMENTS: ORTHOGONAL TRIHEDRAL VERTICES

Sidney Liebes, Jr.

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

A simple analytical procedure is introduced for utilizing a ubiquitous engineering and architural structural subelement to facilitate automatically cuing, monoscopically inferring surface structure and orientation, and resolving stereo correspondences: orthogonal trihedral vertices, or OTVs. OTVs occur in profusion indoors and out. They are identifiable, and are a rich source of information regarding relative surface conformation and orientation. Practical considerations often constrain OTVs to be vertically aligned. General oblique perspective properties of OTVs are examined. The especially important case of nadir-viewing aerial stereophotogrammetry is developed in detail. An object-space vertex labeling convention incorporates vertex type and orientation. A set of image space junction signature rules based upon the object space invariance of OTV edge vanishing points enables unambiguous vertex label assignment for interior and exterior OTVs. An independent application of the labeling scheme to both members of a stereo pair, taken at arbitrarily wide convergence angle, identically labels corresponding juntions. An illustrative example is presented. Algorithmic implementation has not yet been undertaken.

1. Introduction

A basic goal of automated image interpretation is achievement of a symbolic description of the imaged scene. The relatively limited potential for extracting quantitative geometric descriptions from single images is vastly expanded for stereo images. Whereas the present work was motivated by problems in stereo processing, it will be seen to have appicability to single image interpretation as well.

It has been the traditional goal of automated stereo processing systems to attain an accurate and detailed depth map. The central stereo problem is that of establishing correspondences between points in stereo image pairs associated with common object space points in the original scene. Area cross-correlation techiques [Hannah 74, Kelly 77, Panton 78] have worked satisfactorily for smoothly undulating relief, but are unsuitable where surface slope or range are discontinuous, in relatively textureless regions and where interimage surface brightness disparities are extreme. An edge-based approach [Arnold 80, Baker 80, Grimson 80, Henderson 79, Marr 79] complementary to that of area cross-correlation, offers advantages for complex structural configurations.

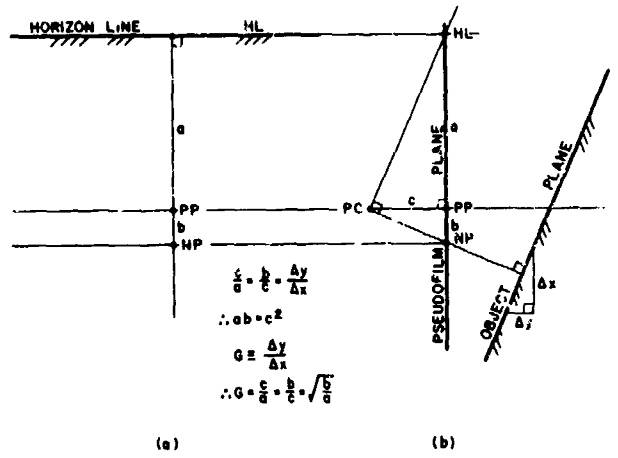
Whereas the goal of symbolic description is generally explicit in single image processing, it has often not been so in stereo image processing. It is to be expected that a general purpose truly powerful stereo image processing system should be capable of delivering not only an accurate and complete depth map, but also a quantitative symbolic description of the imaged scene. The processes of a) resolving stereo correspondences, b) arriving at geometric descriptions of ranged surfaces, c) generating volume descriptions of imaged objects, d) inferring the geometry of portions of structures visible in only a single member of the stereo pair, e) cuing on special features, and f) generally describing the content of the imaged scene all can be facilitated by utilizing world model information relevant to the environment in question. The development of a robust and accurate stereo ranging system and a powerful general purpose quantitative symbolic description capability might be well go hand-in-hand.

Inference plays an important part in human visual perception. Knowledge of the nature of our surroundings influences perceptual processes. Consider a recent experience of the writer. While driving out of the university campus in the dusk and in drizzle one evening, a figure suddenly looming, forward and to the left, was perceived as a bicyclist approaching on collision course from a range of about 50 feet. The "bicyclist" turned out be a smudge on the windshield, abruptly illuminated by the swinging head lights of a car 100 yards ahead. The apparition was processed reflexively, and conservatively, for a fraction of a second as a bicyclist because, presumably, a) it was superposed on a small campus side street from which bicyclists frequently emerge, b) it subtended roughly the proper angular size, c) being stationary in the field of view it was "on collision course", and d) it was a likely time for a bicyclist to be on that road. This is a manifestation of the *Hunter's Principal*, namely, to "shoot at anything that moves" [Binford 81]. The conclusion happens to have been erroneous, but it was a conservative and proper one under the circumstances. Though the above incident was clearly not processed in stereo, it is an example of the human system invoking an elaborate inferential mechanism, doing the best it could with the context, constraints, world model information and the limited quality input available to it.

It seems reasonable to expect that a powerful automated system for generating symbolic descriptions of stereoscopically as well as monoscopically imaged scenes should incorporate capabilities analogous to those listed above. The system should be facile at recognizing or inferring the presense of common or expected features. We develop a case in point below.

2. Othogonal Trihedral Vertices in General Perspective

We wish to consider an application in the practical context of the world of engineering and architectural structures. We are interested in contributing to a versatile wide-angle stereo processing system that can accomplish the processes listed earlier, namely, resolve stereo correspondences, infer surfaces and volumes, infer structure visible in only one image, and have the rudiments of a capability to report a symbolic description. Two kinds of three-dimensional structural feature elements that abound in such a world are the corners of right parallelepipeds (RPPs) and portions of right circular cylinders. We wish here to concentrate on the characteristics of the ubiquitous interior and exterior corner elements of RPPs, which we refer to as *orthogonal trihedral vertices*, or OTVs. Though these particular feature elements are highly specialized, their projected images exhibit rich qualitative structural and orientational information. We adopt the terminology of Waltz [Waltz 72] in referring to object space corners as it vertices and their two-dimensional projections as junctions.



HORIZON LINE
HL
HL
a
a
PP
b
NP
PC
c
PP
b
NP
PLANE
PSEUDOFILM
PLANE
OBJECT
Δx
Δy
$\frac{c}{a} = \frac{b}{c} = \frac{\Delta y}{\Delta x}$
$\therefore ab = c^2$
$G \equiv \frac{\Delta y}{\Delta x}$
$\therefore G = \frac{c}{a} = \frac{b}{c} = \sqrt{\frac{b}{a}}$
(a)
(b)

Figure 1. Perspective camera imaging inclined plane. (a) Image on pseudofilm plane. (b) View perpendicular to camera axis and parallel to object plane.'

The images that we shall be dealing with are assumed to be produced by central projection in a planar perspective camera. An examination of the properties of a perspective image of an inclined plane will lead us to useful observations regarding the projective properties of OTVs, formed by the intersections of orthogonal triples of planes. Figure 1 schematically illustrates a perspective camera photographing an inclined plane. The view in Figure 1b is orthogonal to the axis of the camera, and parallel to the object plane. The camera is shown recording the image on a forward, or *pseudofilm plane* PFP, oriented parallel to the true film plane, and located the same distance in front of the *perspective center* PC as the true film plane is behind it. The image recorded on the pseudofilm plane is identical to that formed on the true film plane, but for a rotation of 180 degrees about the *camera axis*, that passes through the perspective center normal to the film plane. The pseudofilm plane offers the perceptual advantage of exhibiting its image "right side up". Henceforth, we shall use *film plane* to refer to either the pseudofilm plane of the true film plane. The point of intersection of the camera axis with the film plane is referred to as the *principal point* PP. The distance between the perspective center and the principal point is called the *camera constant* c. The point of intersection with the film plane of a ray extended from the perspective center in a direction normal to the object plane is designated the *normal point* NP. We construct a *horizon plane* through the principal point parallel to the object plane. Its intersection with the film plane defines the *horizon line* HL. Note that the product of the distance *a* from the principal point to the horizon line and the distance *b* from the principal point to the normal point is equal to c^2. It will additionally be seen from Figure 1b that the distance *b* is equal to the product *Gc*, where *G* is the magnitude of the gradient of the height of the object plane expressed in camera coordinates. Figure 1a illustrates the perspective image recorded on the film plane. The normal point lies on an extension of a perpendicular from the horizon line through the principal point.

The reader is cautioned that within this paper we do not always take the care that is merited to distinguish in our terminology between object space features and their images. This is not entirely a trivial matter. The writer has more than once been victim of the sloppy thought that can result from a confusion of object space and image space constructs. It would be justified, in a more careful writing, to take care to make the distinction, unless it has been determined that there is no potential for either confusion or error.

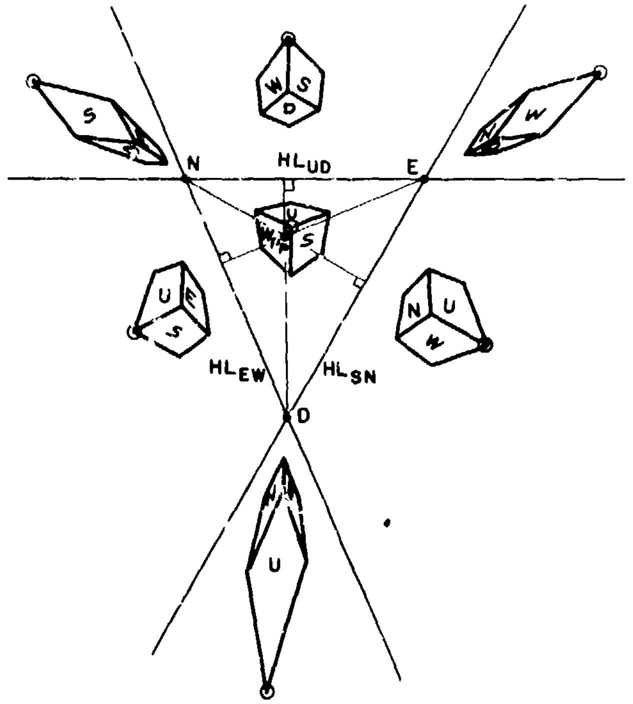


Figure 2. Perspective view of seven RPPs. The RPPs are not identical, but each is aligned with its faces parallel to the corresponding faces of the other RPPs. The three straight lines that intersect to form the central triangle are horizon lines for the faces of the RPP.

Figure 2 illustrates a perspective view of seven RPPs. The RPPs are not identical, but each is aligned with its faces parallel to corresponding faces of the each of the other RPPs. Corners corresponding to the central corner of the central RPP are circled. The point PP is the principal point of the film plane. Consider for the moment the single RPP located inside the central triangular region. Its faces are labled U, W, and S, for "up", "west", and "south", respectively. No particular absolute orientation is implied. The familiar directional associations are to facilitate conceptualization. The horizon line for the face labeled U is designated HL_{UD}, to indicate that it is the horizon line for both the U face and its parallel opposing face D, for "down", as well. Indeed HL_{UD} is the horizon line for the entire family of planes parallel to these two faces of the RPP. The horizon lines for the other faces constitute the remaining sides of the central triangle. The intersections of the three horizon lines correspond to the vanishing points for the edges bounding the faces of the RPP. Now, it is a feature of an OTV, as opposed to the case of an arbitrary object space trihedral vertex, that the normal to any face associated with the vertex is parallel to the edge departing the vertex. Thus there is a coincidence of normal points and edge vanishing points in the perpective plane. It will therefore be appropriate for the case of RPPs to label the vanishing points of the edges with the names of the RPP faces whose outward normals are directed toward them. Thus, the vanishing points for the normals to the RPP faces U, W and S are labled with the names of their opposing faces, namely, D, E and N, respectively. The three faces U, W and S are equivalent in the projection of Figure [illegible] Therefore, the interrelationships involving principal point, horizon lines and vanishing points de-[illegible] Figure 1 apply equally to each of these faces.

The three horizon planes associated with the faces of the RPP divide the forward object space into seven *zones*. The film plane is correspondingly divided into seven *sectors* by the three intersecting horizon lines. An RPP has been placed in each of the zones. It will be noted that when an RPP face is translated across its respective horizon, the reverse side of the face (geometrically if not physically) becomes potentially visible. If, for example, an RPP is translated from beneath the HL_{UD} horizon plane to entirely above it, then the D face becomes visible. An inspection of Figure 2 reveals that all faces of a generally oriented RPP may be made visible by translational displacement, without rotation. It should be noted in passing, however, that for the case of symmetrical angular alignment, where each of the OTV edges is equally inclined relative to the camera axis, these edges will be inclined at an angle of approximately 54.7 degrees to the forward projection of the axis. Any departure from symmetrical alignment will force at least one of the vanishing points to move to an even greater inclination to the axis. Thus, a camera with a 110 degree field of view would be required in order to image even the tightest configuration of all three vanishing points. A 90 degree field of view would be required to capture even two of the vanishing points under the most favorable conditions of alignment, corresponding to the third vanishing point being at infinity. A conventional camera can not therefore be expected to capture more than a single vanishing point in any given image.

3. Detection of OTVs in the General Case

We have already suggested that an OTV is an example of an object that generally presents a challenging wide-angle stereo correspondence problem since the projection into the two images can be quite disparate. The detection of OTVs in a single image is not a totally trivial matter either [Roberts 63], even as a geometrical problem, disregarding the realities of dealing with digitized images of real scenes, wherein one must contend with issues of resolution, noise, shadows, and the like. The problem of OTV detection for the case of general oblique perspective projection has not yet been pursued in detail by the writer. We shall shortly be imposing practical constraints that will simplify the problem for a large and important class of cases. Nevertheless, we reflect briefly on the general case. A necessary condition for a three-legged image junction to correspond to an OTV is that the projections of the legs pass though an acceptable triple of vanishing points, appropriately arrayed about the principal point, per the conditions discussed in conjunction with Figures 1 and 2. Though the condition is not sufficient, it is expected to be highly indicative. There are constraint conditions on the values that can be assumed by the projected internal angles at triple junctions corresponding to OTVs. Though the writer has not yet pursued the point, it is speculated that these are limiting conditions corresponding to those that must be satisfied by the vanishing point relations. When there is coherance of alignment of object space features over a relatively wide camera window, then vanishing point locations can be inferred from single or concurrent intersections of leg extensions, and the vanishing point configuration checked. If it is known in advance or presumed that one is dealing with RPPs, then the junction location and the directions of the film plane projections of the three legs is sufficient to yield the orientation of the OTV relative to the camera, though of course not the range. Stereo correspondence and triangulation could establish the latter.

Most of the OTVs in a given structure will be aligned with their edges mutually parallel. In fact it will not be uncommon for assemblies and collections of stuctures to be similarly aligned, such as, for example, in the case of buildings in a city block. External factors can mediate absolute alignment of OTVs as well. Of these, the direction of the gravity vector is perhaps the single most important determinant in the alignment of cultural objects relative to the Earth. This is of course especially the case for architectural structures. Throughout such structures right angles and OTVs abound. Exterior OTVs are seen, for example, at intersections of walls and roofs of concrete buildings, at doorways, windows, etc. They are found in structural interiors at the corners of rooms, doors, windows and the like, as well as on their contents, such as innumerable small items, machinery, cabinets, shelves and desks. Most of these OTVs will contain vertical edges, and often they are mutually azimuthally aligned as well. Walls are generally vertical, and thus too the edges where they intersect. Roofs are most generally composed of planar sections, the surfaces largely being either horizontal or symmetrically configured shedding slopes the perimeters of which are horizontal.

4. Nadir-Viewing Aerial Stereophotogrammetry

The most common application of stereo processing is in aerial stereophotogrammetry. The nominal situation involves acquisition of a pair of photographs taken by a nadir (vertically downward) directed camera at stations horizontally displaced from one another by a substantial fraction of the height of the camera above the terrain. Such wide-angle stereo favors accuracy, while at the same time complicating the determination of stereo correspondences within highly convoluted or complex structures. Because of the importance of nadir viewing aerial stereophotogrammetry, we now direct our attention to this special case. We will concern ourselves with structures containing vertical and horizontal edges, many of which meet to form vertically aligned OTVs. We shall assume for the purpose of the presentation that the camera is pointed directly downward. In practise, the departure from perfect nadir alignment will require a slight generalization of the implementation. Thus, the camera axis will be parallel to the gravity vector and to the vertical edges of structures. This is a degenerate case of the general oblique configuration illustrated in Figure 2, in that a vanishing point coincides with the principal point. Horizontal surfaces will be parallel to the film plane. This makes the configuration an especially simple one to deal with. The coincidence of the nadir vanishing point with the principal point drives the remaining pair of OTV edge vanishing points to infinite distance in mutually orthogonal directions relative to the principal point on the film plane.

It will help in the following discussion for the unfamiliar reader to be alerted the utility of epipolar lines. Consider a pair of stereo cameras located in known relative positions and orientations. Any plane containing the perspective centers of both cameras is called an epipolar plane. The intersections of an epipolar plane with the two film planes are called epipolar lines. Each object space point has associated with it then an epipolar plane and, therefore, a pair of epipolar lines. It follows that corresponding points in a pair of stereo images must lie upon corresponding epipolar lines. This is clearly a powerful constraint in searching for corresponding stereo image points.

We wish now to consider the projective characteristics of nadir-viewed vertically aligned exterior and interior OTVs. We will develop a set of visibility rules, or junction signatures, that will characterize the appearance of the individual OTVs. Given a junction configuration corresponding to an arbitrarily oriented OTV in one image, the rules will enable the quantitative determination of the appearance of the corresponding junction in the other image, as a function of position along the associated epipolar line. The development of the rules will be facilitated by considering the array of RPP wire models illustrated in Figure 3. The perspective in this figure is vastly exaggerated relative to the typical high altitude situation. This is evident from the fact that the ratio of object space height of a vertical edge, such as that corresponding to (C,C'), to the height of the camera above its base C' is equal to the image space ratio (C,C')/(C,NP). Thus, the camera is in this example not much more than twice the height of the structure. The wire figures will serve as generators for the junction signatures of solid OTVs. The wire models may be considered to rest upon the nominal ground plane, in a common state of rotational alignment about the vertical. The nadir point, horizontal surface normal point and principal point are

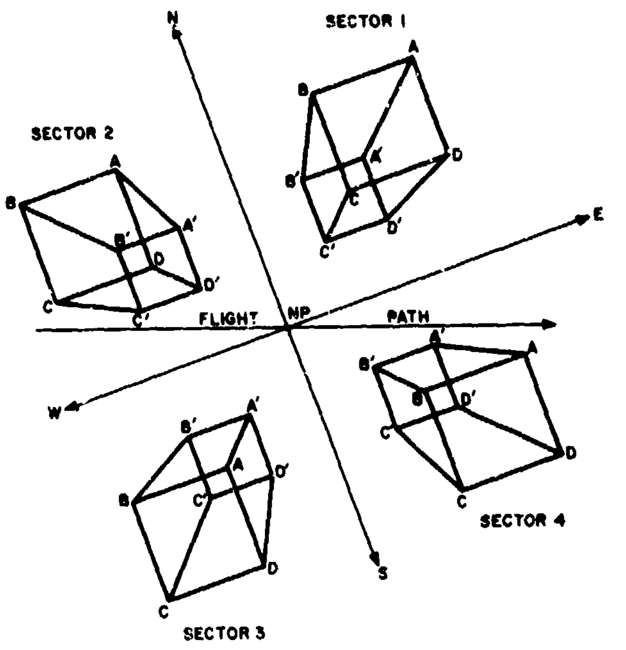


Figure 3. Nadir view of four gravity-aligned wire model RPPs. The horizon lines have been labeled with relative compass directions. The epipolar line through the nadir point corresponds to a projection of the flight path. The wire models are used to generate image plane junction signatures for the OTVs.

mutually coincident at NP. The lines shown intersecting at the nadir point, are the horizon lines for the vertical faces of the model. They are at right angles to one another for this special case. Each of the four wire models has been placed in a separate one of the zones created by the horizon planes, after the fashion of the general oblique case depicted in Figure 2. As per the discussion of the general case, the potential visibility of the sides, vertices and edges of a solid RPP figure filling any one of the wire models is invariant under translation within any given zone. The flight line, which generates the intercamera baseline, extends from left to right.

It will be recalled that the sector demarcation lines in Figure 2 were horizon lines. Horizon lines on the film plane correspond to object-space horizon planes and, therefore, to no particular direction at all in object space. At the risk of some confusion (regarding implications for the general oblique case), it will be a convenience in the nadir case to consider a dual interpretation for the horizon lines on the film plane. We will consider them also to represent projections of object-space lines that pass though a projection of the nadir point, perpendicular to the nadir direction toward the horizontal vanishing points. The rays formed by breaking the lines at the nadir point correspond to the outward surface normals for the vertical faces of the corresponding solid. We label the rays with relative (not necessarily geographic) compass directions N, W, S and E. The E direction is defined to be that of the first leg encountered rotating counterclockwise, looking downward, from the rightward direction along an the epipolar line, labeled in this illustration as the flight path projected through the nadir point. It is tempting to call the sectors into which the horizon lines partition the film plane in the nadir case "quadrants". However, when the camera is not directed precisely downward, the horizon lines will not intersect at right angles on the film plane, as we have observed in Figure 2. Thus, we retain the sector terminology. We number the sectors counterclockwise in the order 1, 2, 3 and 4, with sector 1 corresponding to the region between the E and N legs of the nc; rose. Figure 3 indicates a labeling for the wire model vertices. The four upper vertices are labeled A, B, C and D; the four lower ones A', B', C' and D'. Vertex A' is directly beneath A, and so forth. The labeling assignments must be made in the relative positions indicated with respect to the vanishing point compass rose.

```
WIRE FRAME                         SOLID

    NWSEUD                             NWSEUD
A    XX  X                         A    XX
B     XX X                         B     XX X
C   X  X X                         C   X  X X
D   XX   X                         D   XX   X

    NWSEUD                             NWSEUD
A'   XX X                          A'
B'    XXX                          B'    X X
C'  X  XX                          C'  X  XX
D'  XX  X                          D'   X  X


UP (U) HOLE                        WEST (W) HOLE

    NWSEUD                             NWSEUD
A    XX  X                         A
B     XX                           B     X  X
C   X  X                           C   X    X
D   XX                             D

    NWSEUD                             NWSEUD
A'( XX X )                         A'( XX X )
B'                                 B'    XXX
C'                                 C'  X   X
D'                                 D'


SOUTH (S) HOLE                     The following holas
                                   not visible unless
    NWSEUD                         penetrating, in which
A                                  evont they are
B                                  equivalent to
C      X X                         opposing holes:
D    X   X
                                   NORTH (N)
    NWSEUD                         EAST  (E)
A'( XX X )                         DOWN  (D)
B'
C'     XX
D'  XX  X
```

Figure 4. The set of sector 1 junction signatures for wire model, and for interior and exterior OTVs.

We now introduce a junction-signature assignment for the OTV projections on the film plane. The scheme we employ assigns to each junction the directions of the vanishing points associated with each of the visible outward-bound edges of the object space OTV. For the case of the wire model, the assignment is invariant under translation of model relative to camera, or vice versa, throughout the entire region of object space in front of the camera. Consider for example wire model vertex A. It has associated with it edges directed W, S and D. Each of the edges is visible. Thus, we say that junction corresponding to the wire model vertex A has signature WSD. In similar fashion we assign to wire model vertex D', for example, the junction signature NWU, and so forth for the remainder of the junctions. The signatures for the eight wire model junctions are indicated in the upper left of Figure 4. The eight junction signatures are seen to be unique.

The signatures for a RPP solid may be generated from the wire models of Figure 3 by imagining the wire model edges to correspond to the edges of a RPP solid. The signatures for the junctions associated with each of the solid vertices is a obtained by masking the signatures for the wire model vertices according to the visibility of the associated edges, that is according to whether or not they are self obscured. A solid junction signature will consist of the list of the names of the object space directions associated with the outward pointing visible legs in the projection. The signature associated with a solid vertex will be invariant under vertex translation throughout any given one of the four object space zones into which space is partitioned by the two vertical horizon planes. Abrupt changes of signature can, however, occur upon translation of a vertex across a horizon plane. Thus, it is necessary to indicate a separate set of signatures for each of the four zones within which the OTV can reside, or correspondingly for each of the four sectors into which its projected junction can fall. Consider, for example, the wire model image in sector 1 of Figure 3 to correspond to a solid. Solid vertex A is assigned junction signature WS, compared to the corresponding wire model junction signature WSD, since for the solid the edge directed toward the nadir point is invisible. Vertex B is assigned junction signature SED, the same as that for the wire model, since all edges of this vertex are visible. The junction signatures associated with RPPs situated in other zones are developed in like fashion to that indicated here for sector 1. The set of sector-1 junction signatures associated with the exterior OTVs of a zone-1 solid is indicated in the upper right of Figure 4. Each signature is seen to be unique. Appendix A lists, in condensed form, the complete set of junction signatures associated with solid exterior OTVs occurring in each of the four possible zones. The junction assignments are organized in accordance with the sectors within which they fall. Since the signature of a solid exterior vertex of particular orientation can only change when the vertex is is translated across a horizon plane, the vertex label is completely and uniquely characterized by a) the sector within which it is appears, and b) its junction signature.

Finally, we wish to consider reentrant, or interior OTVs, such as would be encountered, for example, in images of roof depressions, windows, doors and the like. We will refer to these as *interior* OTVs. Their signatures can be generated by a considering the images of rectangular holes in planar solid surfaces. It is again convenient to refer to the wire models of Figure 3. Consider now the wire model in zone 1 to be the junction-signature generator for the vertices associated with a rectangular hole in the U, or top, side of a horizontal surface of a solid. Reference to Figure 3 indicates that an interior OTV at vertex location A will have junction signature WSD, identical to that for the wire model. We assign this vertex the label UA. The vertex at B, labeled UB, will have junction signature SE, compared to SED for the wire model. Vertex A' will have signature WSU, if the hole is shallow enough for the vertex to be visible. A WSU junction signature can be generated by holes in any of the three sides. Furthermore, it is impossible from a consideration of the directions of the legs alone to determine whether the hole is blind or is clear through to the opposing side. This vertex type cannot be uniquely associated with a hole on any particular face. We will label it X' when it is later recognizied in an image. The fact that its visibility is indeterminate, the signature will be enclosed in parentheses. This is the only solid vertex label, among both interior and exterior OTVs, that has multiple junction entries under the present scheme. It speaks more to the nomenclature, however, than to the geometry, since all of these indistinguishable vertices have identically directed edges in object space. The signatures for the junctions situated in other sectors are developed in like fashion to that indicated here for sector 1. The set of junction signatures for sector 1 interior OTVs is presented in the lower portion of Figure 4. The complete list of interior OTV junction signatures for all sectors is given, in condensed form, in Appendix A.

Though both the sector and visible-edge signature assignments may differ in stereo images junction pairs corresponding to the same object space OTV, the labeling assignment, which is determined by the joint values of sector number and signature assignment, will be identical. This will be true for the general oblique case as well as for the nadir case that we have concentrated on. This is the basis for the utility of the scheme for facilitating resolution of stereo corrspondences. Application of the labeling scheme to corresponding stereo junctions yields identical label assignments regardless of the size of the convergence angle. Additionally, the absolute orientation of the OTV is established by the inference of the directions of the vanishing points. Though the labeling assignment will be identical for both members of any given stereo pair, regardless of relative camera orientation and position, this does not imply that the assignment is the same for all pairs of cameras that might record the scene. The assignment is dependent upon the object space direction of the intercamera baseline which, inturn, defines the family of epipolar planes. The direction in space of the baseline, and the stereo sense in which it is viewed, influences assignment of zones and sectors, and the labels of vanishing points and junctions for the members of a stereo pair of images. The labeling assignment is stereo-pair specific. The inherent orientation information is relative to the stereo camera system.

The search for gravity aligned OTVs in nadir imagery is more straightforward than that described for the the general oblique case. It is a feature of planar perspective projection that the projection of a planar object space figure aligned parallel to the plane of projection is but a rescaled version of an orthographic projection of the given figure. Thus, in nadir photography, contour lines and horizontal surfaces are rescaled orthographic projections. Horizontal surfaces of differing heights will experience relative displacement on the image plane. Horizontal angles are invariant under the projection. Nadir aligned OTVs will have two horizontal and one vertical edge. Either two or three of the legs will be visible on projection. If Three legs are visible, then the horizontal pair will project as a right angle and the vertical leg will project as an edge aligned

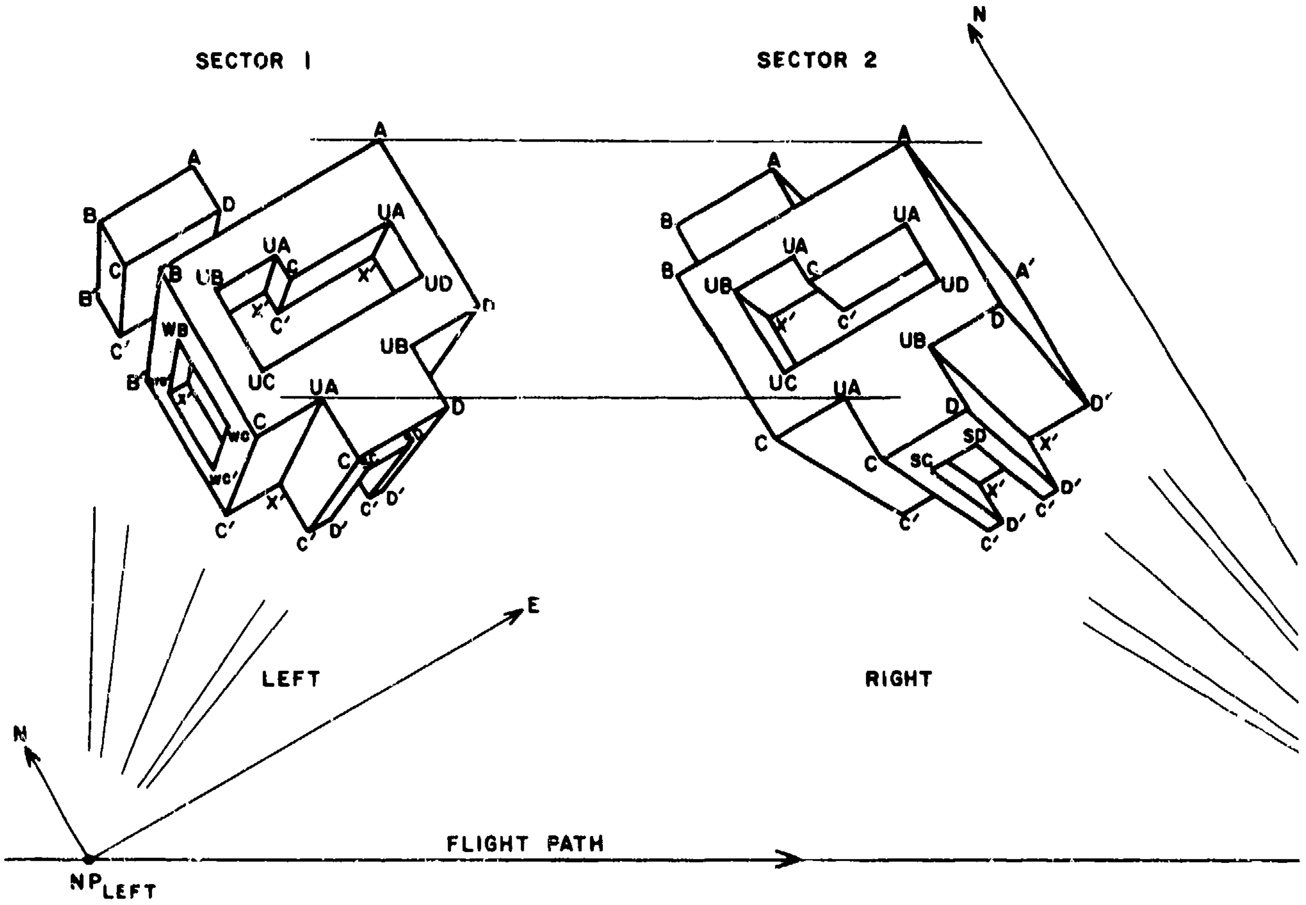


Figure 5. Stereo image pair of "architectural" complex, acquired by nadir viewing cameras. The left image is in sector 1 and the right image is in sector 2. The OTV labels have been assigned by an application of the junction signature tables developed.

with the nadir point. If only two edges are visible, then they will project either as a right angle or as one edge aligned with the nadir point and a second edge generally in some other direction, though parallel with that of other like-aligned OTVs, if equivalent legs of the latter are visible. The three-legged OTV junction appears to be relatively the most unambiguous indicator of an OTV. Local collections of image lines associated with a common triple of appropriately arrayed vanishing points would appear to suggest presense of OTVs. Degeneracies arise along horizon lines, with a double degeneracy occurring at the nadir point. T-juntions, oblique and orthogonal, can suggest painted surfaces, obscuration of features, and the like, as well as a degenerate OTV. A myriad of factors can complicate the detection of OTVs in real imagery, such as signal to noise ratio, resolution, shadow, painted markings, alignment degeneracy, and the like.

An illustrative application of the OTV typing scheme is depicted in Figure 5. The figure schematically depicts a pair of nadir-acquired aerial stereo images of a gravity-aligned "architectural" structural complex. Epipolar lines run left-right. The OTVs are all commonly aligned in azimuth in the figure. Our convention for labeling the compass directions orients E in the direction to the upper right and N in the direction to the upper left in the figure. The scheme places the all the OTVs of the left image into sector 1, and all those in the right image into sector 2. The sector-specific junction signature rules, given in the Appendix A have been used to associate object space labels with the vertices. It will be noted that the corresponding vertices in the two images have been identically labeled.

This research was supported by the Autonomous Terminal Homing Program of ARPA under subcontract C10305 to SRI, International, and by the Image Understanding Program of ARPA under ARPA contract MDA903-80-C-0102 and subcontract HQ80C2530R to Lockheed Missiles and Space.

Acknowledgement

The writer wishes to express his appreciation to David G. Lowe for the use of his TEX protocols for formatting this paper.

References

[Arnold 80] Arnold, R.D. and T. O. Binford, "Geometric Constraints in Stereo Vision," *Proc SPIE,* San Diego, Cal, July 1980.

[Baker 80] Baker, H., "Edge-Based Stereo Correlations," *Image Understanding Workshop,* Univ. Md., April 1980.

[Binford 81] Binford, Thomas O., *(private communication; see also* "Inferring Surfaces from Images," *in press: Artificial Intelligence,* 1981.)

[Hannah 74] Hannah, M. J., "Computer Matching of Areas in Stereo Images," *Artificial Intelligence Laboratory* Stanford University, Memo AIM-239, 1974.

[Grimson 80] Grimson, W. E. L., "Computing Shape Using a Theory of Human Stereo Vision" PhD thesis, Dept. Math., MIT, 1980.

[Henderson 79] Henderson, R. L., W.J.Miller, C.D.Grosch, "Automatic Stereo Reconstruction of Man-Made Targets," *SPIE Proc.,* Huntsville, Aug 1979.

[Kelly 77] Kelly, R., P. McConnell, S. Mildenberger, "The Gestalt Photomapping System," *J. of Photogrammetric Engineering and Remote Sensing,* 1977, 47:1407-1417.

[Marr 79] Marr, D., and T. Poggio, "A Theory of Human Stereo Vision," *Proc. R. Soc. Lond.,* 1979, B 204:301:328.

[Panton 78] Panton, D. J., "A Flexible Approach to Digital Stereo Mapping," *Photogrammetric Engineering and Remote Sensing,* 1978, 44:1499.

[Roberts 63] L. G. Roberts, "Machine perception of Three-Dimensional solids," *Technical Report No. 315, Lincoln Laboratory,* M.I.T. 1963; also in *Optical and Electro-Optical Information Processing,* Tippett, J. T. et al (eds.), MIT Press, Cambridge, Mass., 1965, pp. 159-197.

[Waltz 72] Waltz,D., "Generating Semantic Descriptions from Drawings of Scenes with Shadows," *MIT-AI Tech. Rept. AI-TR-271,* 1972; see also, "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Perception,,* P. H. Winston, (ed.), McGraw-Hill, 1975.

Appendix A

Condensed Summary of Nadir-Viewed Gravity-Aligned OTVs

The following is a condensed summary of the solid interior and exterior signatures for all sectors in the case of nadir-viewed gravity aligned OTVs. The four-fold rotational symmetry enables the condensed form. The diagonal numbers are sector numbers. To the right and below each sector number are the associated vanishing point directions and vertex labels, respectively.

```
      SOLID                   UP HOLE (U)        Following holes
                                                 not visible unless
  4     ENWS              4     ENWS             penetrating, in
   3    SENW               3    SENW             which event they
    2   WSEN                2   WSEN             are equivalent to
     1  NWSEUD               1  NWSEUD           opposing holes:

  DCBA   XX               DCBA    XX  X                 HOLE
  ADCB    XX X            ADCB     XX                  FACING
  BADC  X  X X            BADC  X  X                  DIRECTION
  CBAD  XX   X            CBAD  XX               SECTOR
                                                   1    N or E
  prime                   prime                    2    W or N
  DCBA                    DCBA  ( XX X )           3    S or W
  ADCB    X X             ADCB                     4    E or S
  BADC  X  XX             BADC                    ALL   D
  CBAD   X  X             CBAD

 SECTOR  HOLE             SECTOR  HOLE

   1    SOUTH (S)           1    WEST  (W)
   2    EAST  (E)           2    SOUTH (S)
   3    NORTH (N)           3    EAST  (E)
   4    WEST  (W)           4    NORTH (N)

 4      ENWS              4      ENWS
  3     SENW               3     SENW
   2    WSEN                2    WSEN
    1   NWSEAT               1   NWSEAT

 DCBA                     DCBA
 ADCB                     ADCB     X  X
 BADC       X X           BADC  X     X
 CBAD    X    X           CBAD

 prime                    prime
 DCBA  ( XX X )           DCBA  ( XX X )
 ADCB                     ADCB     XXX
 BADC       XX            BADC  X    X
 CBAD   XX  X             CBAD
```

A Procedure for Camera Calibration with Image Sequences

Kenneth L. Clarkson

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, CA 94305

Abstract

A procedure for calibration of the stereo camera transform is described which uses a variable projection minimization algorithm, applied to an error function whose dependence on the five camera model parameters is separable into linear and non-linear components. The result is a non-linear minimization over three variables rather than five. The procedure has been implemented in MACLISP, with good preliminary results.

Introduction

This paper describes a partial solution to the following problem: Images from two cameras are given, with matchings between points in each image which are thought to correspond to the same point in 3-space, with no further information such as the location of known object points. The problem is to use the matchings to find the location and orientation of one camera with respect to the other. The solution described here uses geometrical constraints from the matchings to find a least squares error function of the five parameters involved. This error function is in effect linear in the location parameters, so that best least squares values for those parameters can be obtained for given orientation parameters. The error of this fit is then minimized over the orientation. The result is a non-linear minimization in three variables rather than five, an improvement over previous methods. (cf. [5]) Note that the images are arbitrary, in the sense that they can be from two cameras at one time, or from one moving camera, or from one fixed camera on a moving object.

Derivation of the Procedure

More precisely, let camera 1 be at the origin, looking along the z axis, and let camera 2 be at point c; looking in such a direction that a point y is seen by camera 2 at $R(y-c)$, where R is a rotation (hence orthonormal) matrix. Now we know that the following vectors are co-planar: that from camera 1 to y, from camera 1 to camera 2, and from camera 2 to y. (See Figure 1) These are just y, c, and $y-c$, respectively. Therefore, $y \times (y-c)$ is perpendicular to c. Thus if we know R, and have vectors a parallel to y, b parallel to $R(y-c)$, we have $a \times R^T b$ perpendicular to c. This is true for all object points y, and the image point data (together with camera focal lengths) give a set of vectors a_i, and b_i corresponding to some y_i. These allow us to estimate the location c of camera 2 as the vector minimizing

$$\sum_{i=1}^{m} (c \cdot (a_i \times R^T b_i))^2 \qquad (1)$$

for m the number of image point matchings. Any vector parallel to c will do, so we will fix the third component of c to 1. As a result we have a linear least squares problem in c, given the rotation matrix R of the orientation of camera 2. So we can estimate R by finding a value of it minimizing the error in the fit.

A well known algorithm (among numerical analysts) for solving separable least squares problems of this kind is the variable projection method[1][2]. This method uses the Levenberg-Marquardt iteration (which has the Gauss-Newton iteration as a special case) for the solution of the non-linear part, which involves the

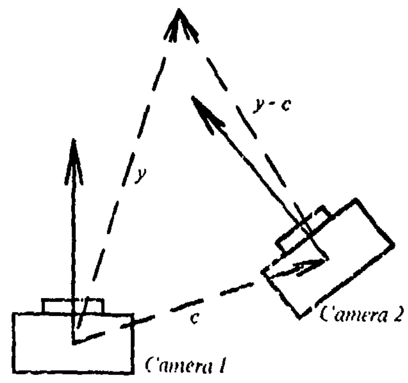


Figure 1. Camera views of object at y

linearisation of the problem at each iteration and the least squares solution of the linearisation. Thus each iteration involves two least squares problems, which can both be solved by using the numerically stable technique of Householder transformations to find the generalised inverses of the matrices involved. The following development of the method as applied to camera calibration follows the discussion in [2].

Let $d_i = a_i \times R(\beta)^T b_i$, where a_i and b_i are unit vectors pointed from camera 1 and camera 2 to y, as easily obtained from the image point matchings. The dependence of R on the three orientation parameters $\beta = (\beta_1, \beta_2, \beta_3)$ is explicitly indicated. (The nature of this dependence is discussed below.) Let $\Phi(\beta)$ be the $m \times 2$ matrix whose i'th row is the x and y components of d_i, and let $y(\beta)$ be the m-length column vector whose i'th component is the z component of d_i. Then the problem of minimizing (1) is equivalent to minimising

$$\|y(\beta) - \Phi(\beta)x\|^2 \tag{2}$$

over $x \in \mathbb{R}^2$, for given β. This can be done in the following way: compute the "orthogonal decomposition" of Φ, i.e., find an $m \times m$ matrix $Q(\beta)$ such that

$$Q(\beta)\Phi(\beta) = \begin{pmatrix} U(\beta) \\ 0 \end{pmatrix} \tag{3}$$

where U is a right triangular 2×2 matrix, and Q is orthonormal. Partition Q into two sub-matrices Q_1 and Q_2, with the first 2 rows of Q forming Q_1 and the remaining $m-2$ rows forming Q_2. Since Q is orthonormal, we know that $\|Qz\| = \|z\|$ for m-vector z, so

$$\begin{aligned} \|y - \Phi x\|^2 &= \|Q(y - \Phi x)\|^2 \\ &= \left\| \begin{pmatrix} Q_1 y \\ Q_2 y \end{pmatrix} - \begin{pmatrix} U \\ 0 \end{pmatrix} x \right\|^2 \\ &= \|Q_1 y - Ux\|^2 + \|Q_2 y\|^2 \end{aligned}$$

Since $\|Q_2 y\|^2$ doesn't depend on x, the optimal x for fixed β is $x = U^{-1}(\beta)Q_1(\beta)y(\beta)$, and we want to minimize $\|f(\beta)\|^2$, where f is the $(m-2)$-vector valued function $Q_2(\beta)y(\beta)$.

We can minimise the norm of f using the Levenberg-Marquardt algorithm, which iteratively refines β as follows: for the current value of β, compute the Jacobian of f for β, the $(m-2) \times 3$ matrix $\{J_{ij}\}$, where

$$J_{ij} = \frac{\partial f_i}{\partial \beta_j},$$

then find the least squares solution of

$$\begin{pmatrix} J \\ \nu I \end{pmatrix} \delta = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{4}$$

where the length of the correction vector δ is controlled by the Marquardt parameter ν, whose value may vary with the iterations. The correction vector δ is then subtracted from β for a new estimate. When $\nu = 0$, this is the Gauss-Newton method. Finding the least squares solution for (4) can be done using the same orthogonal decomposition method as for (2).

We need to compute the Jacobian of $f(\beta)$. The columns of this matrix are

$$\frac{\partial(Q_2(\beta)y(\beta))}{\partial \beta_j} = \frac{\partial Q_2(\beta)}{\partial \beta_j} y(\beta) - Q_2(\beta)\frac{\partial y(\beta)}{\partial \beta_j}. \tag{5}$$

From (3) we know that $Q_2(\beta)\Phi(\beta) = 0$, so that

$$\frac{\partial Q_2(\beta)}{\partial \beta_j}\Phi(\beta) = -Q_2(\beta)\frac{\partial \Phi(\beta)}{\partial \beta_j},$$

and hence approximately

$$\frac{\partial Q_2(\beta)}{\partial \beta_j} \approx -Q_2(\beta)\frac{\partial \Phi(\beta)}{\partial \beta_j}\Phi^+(\beta),$$

where $\Phi^+ = U^{-1}Q_1$, the "generalised inverse" of Φ, as above. This simplification is due to Kaufman[3].

It remains to determine the derivatives $\partial\Phi(\beta)/\partial\beta_j$ and $\partial y(\beta)/\partial\beta_j$. Since Φ and y are obtained from the vectors d_i, we need to find

$$\begin{aligned} \frac{\partial d_i}{\partial \beta_j} &= \frac{\partial a_i \times R(\beta) b_i}{\partial \beta_j} \\ &= a_i \times \frac{\partial R}{\beta_j} b_i \end{aligned}$$

There are many ways to define a rotation matrix using three parameters: we will factor R into a product of rotations R_1, R_2, and R_3 about the x, y, and z axes, rotating β_1, β_2, and β_3 radians, respectively. These matrices have a simple form, for example

$$R_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta_1 & -\sin\beta_1 \\ 0 & \sin\beta_1 & \cos\beta_1 \end{pmatrix},$$

and $\partial R/\partial\beta_1 = (\partial R_1/\partial\beta_1)R_2R_3$, and so on.

Conclusions

The method has been implemented in MACLISP using the techniques for orthogonal decomposition presented in [4]. Preliminary computational experience indicates rapid (10-15 iterations) convergence to machine accuracy when the algorithm converges. Convergence is not global for this class of algorithms, unfortunately, and good estimates (within 0.3 radians, roughly) of the components of β are necessary.

Further work might include the use of confidence weights for the image point matchings, initial rough estimators for β, and the use of convergence acceleration techniques. For completeness, a corresponding procedure should be included for the case that the cameras are at the same location: this implies that $c = 0$, and minimization of (1) isn't meaningful. Another problem is that no translations along the z axis are allowed by this formulation.

Finally, it should be noted that this procedure does not facilitate the use of prior knowledge of the translation, so that in certain situations a full five-variable minimization might be appropriate.

Bibliography

[1] G.H. Golub and V. Pereyra, "The differentiation of pseudo-inverse and nonlinear least squares problems whose variables separate," SINUM 10(1973), pp.413-432.

[2] G.H. Golub and R.J. LeVeque, "Extensions and uses of the variable projection for solving nonlinear least squares problems," *Proceedings of the 1979 Army Numerical Analysis and Computers Conference*, ARO Report 79-3.

[3] L. Kaufman, "A variable projection method for solving separable nonlinear least squares problems," CACM 17(1974), pp.167-169.

[4] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, Prentice Hall, Englewood Cliffs, 1974.

[5] D.B. Gennery, "Modelling the environment of an exploring vehicle by means of stereo vision," Stanford Report STAN-CS-80-805, Stanford University, 1980.

EVALUATION AND REAL-TIME IMPLEMENTATION OF IMAGE UNDERSTANDING ALGORITHMS

Bruce J. Schachter
Glenn E. Tisdale

Westinghouse Electric Corp., Systems Development Division
Baltimore, Maryland 21203

ABSTRACT

In the third phase of its association with the University of Maryland in the DARPA Image Understanding Program, Westinghouse is investigating performance problems with image segmentation algorithms, application of a knowledge base to image recognition, and the use of optical flow techniques. These choices are based upon an anticipated demand for improved recognition performance for weapon delivery and reconnaissance systems. This paper discusses current efforts in each of these areas, as well as the preparation of a comprehensive data base and the conduction of meaningful performance tests.

1. Introduction

The Westinghouse Systems Development Division is now entering the third phase of its association with the University of Maryland Computer Vision Laboratory in the DARPA Image Understanding Program. The Maryland program director is Prof. Azriel Rosenfeld. The Army program monitor is Dr. George Jones of the Night Vision and Electro-Optical Laboratory, Ft. Belvoir.

Highlights of the first phase of the Westinghouse program (1976-78) were the demonstration of a CCD histogrammer-sorter which incorporated a special-purpose Westinghouse CCD chip; and the preliminary design of an entire automatic cueing system, using CCD architecture, in a 3" x 3" x 6" volume (1). During the second phase (1978-80) attention was directed toward the hardware implementation of relaxation algorithms (2,3,4). A non-linear prefilter based upon this work has been demonstrated, and will be incorporated into the AUTO-Q processor (5), as a replacement for conventional averaging filters. Additional effort in this phase was directed toward the use of array processors for algorithm tests where large data bases were involved (4). The current phase of the program began in late 1980. It will expand earlier efforts in the evaluation and real-time implementation of image understanding algorithms. The specific areas for this effort are:

- Investigation of performance problems with image segmentation algorithms;
- Construction of a knowledge base to improve object labeling;
- Development of optical flow techniques.

These problem areas are considered crucial to the success of future image recognition programs which support reconnaissance and weapon delivery operations. The reasons for this assessment will be discussed below.

At the current state of the art, a variety of segmentation algorithms are available which perform target extraction quite well with "clean" imagery, but which deteriorate rapidly with the appearance of noise, clutter, or partial target obscuration. This is a severe bottleneck in the image recognition process. Fortunately, some promising new segmentation algorithms are beginning to appear. Several candidate algorithms will be selected from those developed at the University of Maryland, and other IU programs. One goal of the current effort is the development of meaningful statistical tests to evaluate their performance. With this objective in mind, a data base has been compiled which reflects complex reconnaissance and weapon delivery scenarios.

The belief that a knowledge base may be useful in image recoginition is based upon the inability of machines to deal with some complex scenes which humans can interpret by using memory, context, and reasoning. This idea has been around for years, but with very little successful implementation. New optical flow techniques offer some promise of success in several areas. They can potentially assist in the (passive) determination of range to a target, the detection of target motion, and the evaluation of sensor line of sight changes. The following paragraphs describe the work which has been initiated in each of the above areas, the selection of a "realistic" data base, and test plans.

2. Image Segmentation

Image segmentation is a process of partitioning an image into regions--each having different properties. The class of images within this project's scope are FLIR target/background scenes. Thus, segmentation can be regarded as a process of separating targets from background clutter.

The standard method of segmenting an image is by gray level thresholding. Here the classes correspond to gray level ranges, e.g. "light-hot" and "dark-cool". Since these ranges are not known in advance, they must be determined by examining the gray level histogram and looking for peaks (one dimensional clusters), and choosing thresholds (one dimensional decision surfaces) that separate the peaks.

A number of investigators have suggested that multidimensional feature space should also be useful for segmenting complex gray scale images. A variety of features may be defined over a neighborhood set about a pixel, e.g. mean, median, variance, commonality, total variation (6). This approach could be employed when a single feature, such as gray level, is not adequate for segmentation because the given image contains a number of textured regions whose gray level ranges overlap.

Initial work at Westinghouse has indicated that thresholding by cluster detection is not adequate for separating targets from background. Gray level target and background clusters are often not separable, i.e. their probability densities overlap. Likewise, the response of local operators tends to be rather variable, not yielding well defined clusters. The basic weakness of segmentation schemes which use only local feature values is that they attempt to classify image parts without regard to their relative positions in the image. It should not surprise us that any approach which does not take spatial contiguity fully into account fails much of the time.

The segmentation algorithms investigated in our study are those that make use not only of similarity but also of proximity. The candidate algorithms for our study include, but are not limited to, the following:

- Superslice (7)
- Pyramid spot detector (8)
- Pyramid linking (9)
- Two-label relaxation (10,11)
- Spoke detector/segmentor (12)

A data base of 50 FLIR images (128 x 128) has been assembled from Army, Navy, Air Force and Westinghouse sources. Several images from this data base are shown in Figure 1. Each of the candidate algorithms will be tested on the first ten images. Those that perform well will then be tested on the remainder of the data base.

This brings up the question of how to evaluate the performance of an algorithm. Several approaches are being considered for use alone or in combination. One approach is to have a human "hand segment" the images. The resulting binary image plane then becomes an estimate of the ground truth. Another approach is to use synthetically generated images for which the ground truth is known by construction. One such image has been included in our data base. A third approach is to feed the output of each of the segmentors into a common classifier. The classification accuracy is then assumed to give an indication of segmentation accuracy.

The task of comparing a number of segmentation algorithms is by no means clear-cut. The performance of each algorithm is related to the type of noise cleaning done before or after each stage of its operation. Furthermore, each algorithm has one or more parameters which must be adjusted. The optimal performance of an algorithm can only be achieved by fine-tuning these parameters with respect to the class of images under consideration. The robustness of this tuning operation is an extremely important consideration in a military context, but also one which is difficult to evaluate.

Most segmentation algorithms contain a number of processing steps which run in sequence. It may be possible to separate the stages of operation of the various algorithms and combine them in different permutations. Presumably, by careful analysis, one could take the best parts of the best of the algorithms and assemble them into a new algorithm. This idea will be investigated further as our study progresses.

3. Construction of a Knowledge Base

The AI approach to scene analysis involves the construction of a knowledge base and the exploitation of constraints implied therein. Certain knowledge about the physical world can always be used. To quote Marr (13):

(C1) "A given point on a physical surface has a unique position in space at any point in time.

(C2) Matter is cohesive; it is separated into objects; and the surfaces of objects are generally smooth compared with their distance from the viewer."

These constraints apply to location on a physical surface. One approach to using such constraints is to develop a primitive scene description and then resort to a convergence of evidence. This primitive description can take the form of lines, edges, corners, blobs (obtained from segmentation), tilt of the ground plane and location of the horizon.

Higher level information can be incorporated at a later stage. Higher level knowledge deals with the particular goals of the analysis and domain of the data. For example, tanks sometimes leave warm dust trails or tread tracks. Trucks and jeeps often travel over roads. Targets tend to cluster into groups. An object floating above the ground is more likely to be a helicopter than a tank.

More information is contained in a sequence of images than in a single snapshot. The motion of features derived from the perspective projection of a scene onto a view window is called the scene's optical flow. The optical flow results from a combination of the view window's movement through the 3-D environment and the movement of scene components within the environment. Westinghouse is using data extracted from optical flows to provide clues to local surface orientation, relative motion, and depth relationships. An optical flow image produced by Westinghouse's hardware system is shown in Figure 2.

Another source of information takes the form of a partial world model which can be developed and stored off-line and retrieved when needed. One good source is the Defense Mapping Agency's digital culture and terrain elevation files. If the location and attitude of an observer are known, then this data can be used to construct an initial model for his 3-D environment. Other data of this type include the planned flight path, weather conditions, and gathered intelligence.

REFERENCES

1. G. E. Tisdale and T. J. Willett, Hardware implementation of a smart sensor: a review, in Proc. of the Image Understanding Workshop, (Cambridge, MA, May 3-4, 1978), pp. 1-8.

2. T. J. Willett, Hardware implementation of image processing using overlays: relaxation, in Proc. of the Image Understanding Workshop, (Pittsburgh, Nov. 14-15, 1978), pp. 175-181.

3. C. W. Brooks, G. E. Tisdale, and T. J. Willett, Relaxation, stolic arrays, and universal arrays, in Proc. of the Image Understanding Workshop, (Palo Alto, CA, April 24-25, 1979), pp. 164-170.

4. A. R. Helland, G. E. Tisdale and T. J. Willett, Higher level algorithms, evaluation and implementation, in Proc. of the Image Understanding Workshop, (Los Angeles, Nov. 7-8, 1979), pp. 15-23.

5. A. R. Helland, Convergence properties of two label relaxation, in Proc. of the Image Understanding Workshop, (College Park, MD, April 30, 1980), pp. 176-181.

6. B. J. Schachter, L. S. Davis and A. Rosenfeld, some experiments in image segmentation by clustering of local feature values, Pattern Recognition 11, 1979, pp. 19-28.

7. D. L. Milgram, Region extraction using convergent evidence, in Proc. of the Image Understanding Workshop, (Minneapolis, April 20, 1977), pp. 58-64.

8. M. Shneier, Using pyramids to define local thresholds for blob detection, in Proc. of the Image Understanding Workshop, (Los Angeles, Nov. 7-8, 1979), pp. 31-35.

9. A. Rosenfeld, Some uses of pyramids in image processing and segmentation, in Proc. of the Image Understanding Workshop, (College Park, MD, April 1980), pp. 112-115.

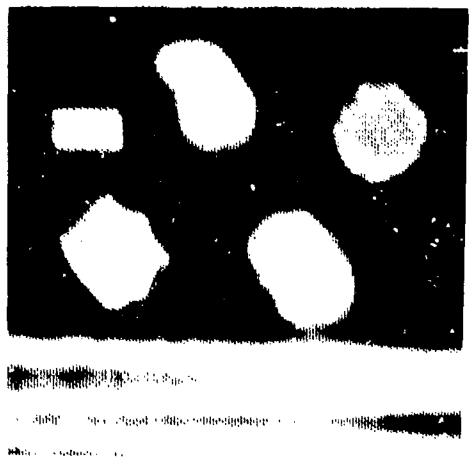
10. A. Rosenfeld, R. A. Hummel, and S. W. Zucker, Scene labeling by relaxation operations, IEEE Trans. Systems, Man, and Cybernetics 6, June 1976, pp. 420-433.

11. D. P. O'Leary and S. Peleg, Analysis of relaxation processes: the two-node, two-label case; University of Maryland, Computer Science Dept., T.R.-967, College Park, MD, Nov. 1980.

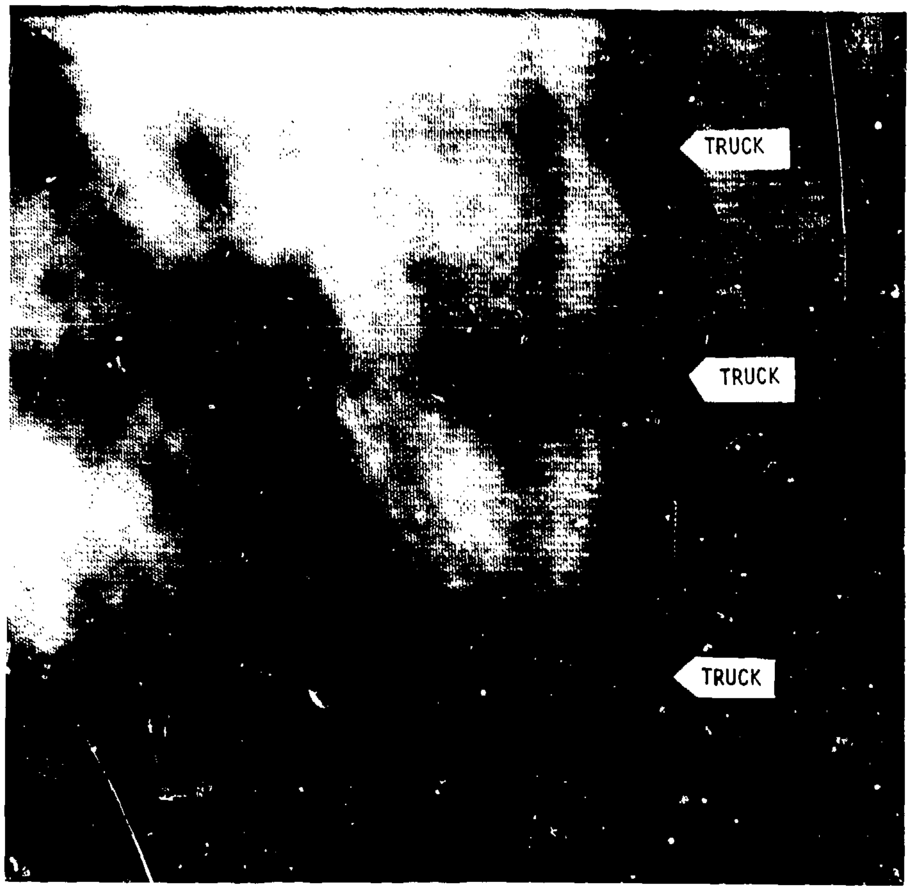
12. L. G. Minor and J. Sklansky, The detection and segmentation of blobs in infrared imagery, to appear in IEEE Systems, Man, and Cybernetics.

13. D. Marr and T. Poggio, Cooperative computation of stereo disparity, M.I.T. Artificial Intelligence Lab, AI Memo 364, Cambridge, MA, June 1976.

14. G. E. Tisdale, The AUTO-Q Digital Image Processor for target acquisition and handoff; presented at IRIS 1980 Conference, Seattle, WA, May 1980.

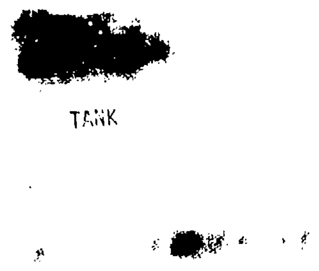


(a)

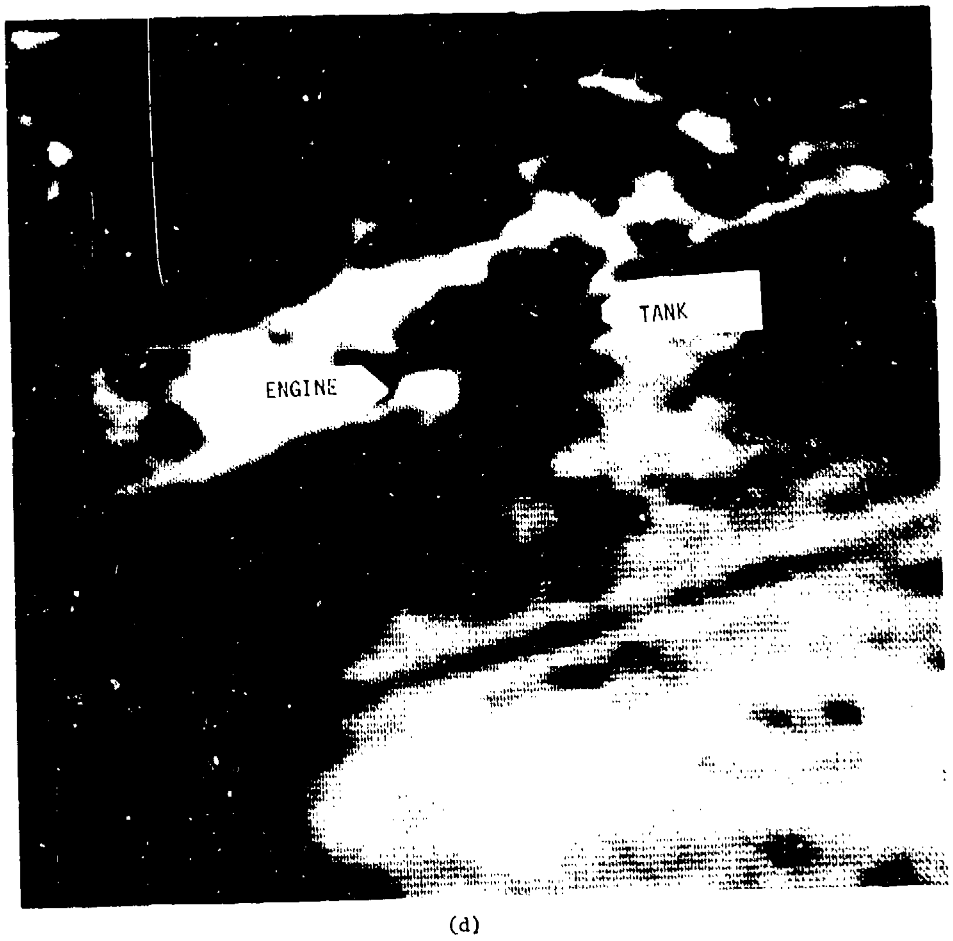


(b)

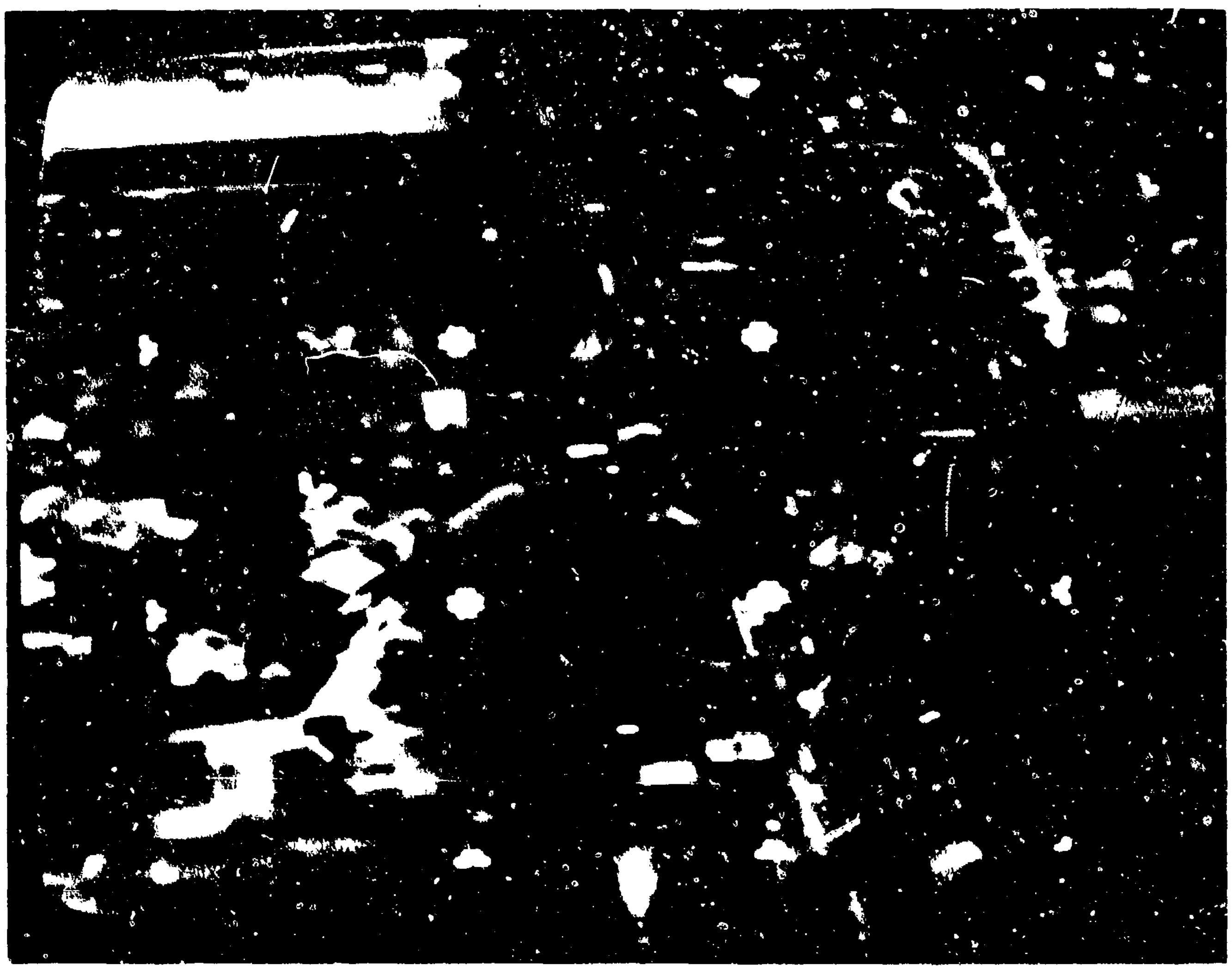
Figure 1. Part of Test Data Base. Photographs are (a) simple test image, and FLIR scenes from (b) Navy, (c) Army, and (d) Air Force sources.



(c)



(d)



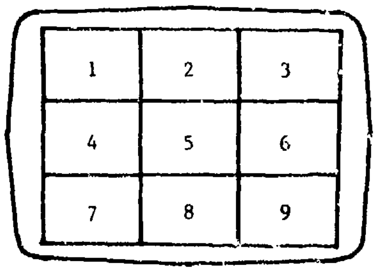


Figure 2. Optical Flow Scene Produced by Westinghouse AUTO-Q Digital Image Processor (14). The input to the device was from a TV camera which was rotated between successive frames. The optical flow image consists of 9 windows, whose corners are marked. Each window contains a single flow vector.

A GENERAL PURPOSE VLSI CHIP FOR COMPUTER VISION WITH FAULT-TOLERANT HARDWARE

Michael R. Lowry and Allan Miller

Artificial Intelligence Laboratory
Stanford University, Stanford, Ca. 94305 *USA*

Abstract

This article describes a VLSI NMOS chip suitable for parallel implementation of computer vision algorithms. The chip contains a two dimensional array of processors, each connected to its four neighbors. Each processor currently has 32 bits of internal storage in three shift registers, and can do arbitrary boolean functions as well as serial bit arithmetic.

Our objective is to make a vision processor with one processor for each pixel. This will require a very high density VLSI implementation, filling an entire wafer. We will need fault-tolerant hardware to deal with the fabrication errors present in such large circuits. We plan to do this by incorporating redundant links in the processor interconnections and routing the links around faulty processors.

Current work focuses on testing a prototype chip with one processor, redesigning the chip for a more compact and regular layout, and designing the redundant link interconnections and hardware support for picture size arrays of processors.

Introduction

Many computer vision algorithms consist of repeating the same operation over each local region of an image. This can take excessive amounts of time on a serial machine, even for non-production research purposes. Current vision algorithms have reached a limit due to the processing power of the computers on which they are running. Further improvements in computer vision will need a fast, general purpose array processor.

To achieve greater speed, we are developing a high density two-dimensional array of general purpose processors. All processors execute the same instruction sequence, controlled by a common microcode bus, but individual processors can be selectively enabled by data-dependent conditions. This idea is not new, but past work has been limited to small arrays because of low-density and high-cost hardware [1,2]

In a sense, we are trying to achieve the same capability as human visual processing of doing all local operations in parallel [3]. Our approach differs somewhat from other current work in VLSI computer vision [4] in that we are developing general purpose hardware, as opposed to hardware for a particular operation such as convolution. Since most image processing consists of a sequence of linear and non-linear operations, even for a fixed system a special purpose chip is suitable for only part of a total system. For development and testing of new algorithms general purpose hardware is needed.

Processor Design

Figure 1 shows the data path of one processor. It is important to note that the processor is serial; in other words, each line in the diagram represents a single connection. The R register is 16 bits, and the D1 and D2 registers are each 8 bits. MUX1 and MUX2 select the inputs to the adder. The adder inputs can be selected from R shift output, D1 shift output, D2 shift output, zero, one, and the latched adder output of any of the cell's four neighbors. The output selector selects the shift inputs to R, D1, and D2. The output selector is constructed in such a way that one of the three registers can be loaded from the adder output, and the other two registers are loaded from their own shift outputs. In this way, D1 and D2 can be summed into R without losing the contents of either D register.

The control path of the processor is quite simple. There are twelve externally generated microcode bits. Six of the bits select the adder inputs through MUX1 and MUX2. Two of the bits control the output selector: either all three registers are loaded from their shift outputs or one of the three registers is loaded from the adder output and the other two are loaded from their shift outputs. Three of the bits independently enable shifting of R, D1, and D2. The final microcode bit causes the shift enable to be loaded from the adder output. When the shift enable is a zero, register shifting is disabled regardless of the state of the microcode bits. This effectively disables the processor.

Programming examples

Although the processor design is quite simple, it allows several fairly important vision algorithms to be done relatively quickly due to its parallel implementation of local operations.

To convolve an image with a fixed mask, we store the greyscale value of each pixel in the D1 register of one processor and use the method of adding multiples of a spatially shifted image to accumulate the result in the R registers of the processors. The shifted image is put into the D2 register by selecting the proper neighbor as one adder input and zero as another adder input, then doing eight shifts while saving the result in D2. This shift operation can be repeated any number of times in any direction to produce a properly shifted image in D2. One convolution step is then completed by adding the appropriate multiple of D2 to R. For example, if 5 times the value of D2 is to be added to R, first D2 is added to R, then R is arithmetically shifted right two bits, then D2 is added to R again. This convolution step is then repeated for the entire fixed mask.

Using the shift enable, it is also relatively simple to multiply D1 by D2 and save the result in R. R is cleared, then D1 is repeatedly added to R, arithmetically shifting R right one bit between additions. However, just before each add operation, the next bit in D2 is loaded into the shift enable. This has the effect of only adding shifted versions of D1 to R where the corresponding bit in D2 is a one. The result is the desired multiplication, yielding the cross correlation of an image stored in D1 with the image stored in D2.

Thresholding a picture is quite easy, since it can be done by subtracting the threshold from the data using two's complement arithmetic. The final carry can be loaded into the shift enable and any threshold-dependent operation can then be done.

To find directional zero crossings, each processor retrieves the sign bit of the data in its neighbor in the desired direction. Processors with positive data are then enabled by loading the complement of the sign bit into the shift enable (complementing a bit is done by adding one to it after clearing the carry). Each enabled processor then loads its neighbor's sign bit into the shift enable. The result is that only processors that were originally next to zero crossings with negative slopes are enabled (the positive-slope zero crossings can be found by operating in the opposite direction).

By using various parts of the D registers for mantissas and exponents, it is possible to simulate limited floating-point operations, although it is clear to us that eight bits of data severely limits both the range and precision of these numbers.

Although we have not yet investigated further algorithms to the same detail, it seems clear that relaxation methods such as finite element analysis for solving 2-dimensional differential equations are particularly well-suited for local processing. In addition, we are currently investigating a halftoning algorithm that spreads errors due to thresholding in a uniform manner rather than a biased fashion as done in current algorithms [5]. We feel that these diverse applications of our processor are a good indication of its generality.

Technological considerations

Figure 2 shows the processor layout. Our processor is currently approximately square, measuring 500 λ on a side. Since our test chip was fabricated with a 2.5 micron λ, our test processor is 1.25 mm on a side. To make a 512 by 512 array of processors by simply replicating this processor would require a wafer 64 cm on a side. A reasonable design will have to reduce the linear dimension of the processor by 10, or the area by 100.

Current fabrication techniques can achieve a 1 micron λ, giving us a factor of 2.5 linearly, or 6.25 in area. The microcode lines take up approximately one third of the area of our processor, so sharing them between processors gives another factor of 1.2 in area. The memory in the cell can probably be reduced to at least three fourths of its current size (using standard memory cell designs). Since the memory currently takes about one third of the cell, this gives another factor of 1.1 in area. Our original design left much blank area in the interconnection of the processor elements, and the processor could probably be reduced to two thirds of its current size. Since the processor takes up one third of the cell area, a cleaner, more regular processor design would give yet another factor of about 1.13 in area. Taken together, these factors mean that a 512 by 512 processor array is within about a factor of 11 in area or 3.3 in linear dimension of being feasible. We expect to be able to make a 128 by 128 array in the near future, and move toward larger arrays as fabrication technology improves.

The biggest problem with making a wafer-sized chip as we plan to do is overcoming the problem of low yield. The largest commercially available chip today is the Motorola M68000, which is about 6 mm by 7 mm. Since we plan on making a chip with 100 times the area, and the standard yield model decreases exponentially with area, we must use a processor interconnect scheme that can tolerate errors in chip fabrication. [Specifically, the model incorporates an "error density" that is constant over the wafer. If p is the probability of finding no errors in one unit area, then the probability of finding no errors in an area of size A is p^A, so if the area increases by 100 the new yield is simply the old yield raised to the 99th power. If the old yield was 10% the new yield will be 10^{-99}.] We plan to deal with fabrication errors by incorporating extra interprocessor connections in the chip. A test run on the chip will pinpoint bad processors, then in actual use some of the extra interconnections will be removed, leaving a network of good processors.

Redundant element methods were used in early 4K RAMs, where each half of the circuit would be tested independently and chips with one bad memory array would be sold as 2K RAMs. More sophisticated methods are being used today in larger memory chips to delete single rows and columns of bit-storage arrays [6].

We foresee three major issues in making a working fault-tolerant design.

We will need to know more about the statistics of chip errors in order to have a reasonable model on which to base yield calculations and interconnection schemes. Although some data is available, it is mostly statistical in nature and is based on tests of memory arrays. Industrial integrated circuit manufacturers are reluctant to reveal yield statistics of their fabrication facilities, since their profits are dependent on these figures. We plan to design a circuit that can address areas on its surface and check for various kinds of fabrication defects. In this way, we will have a better model of fabrication errors.

It will be necessary to have a redundant inter-

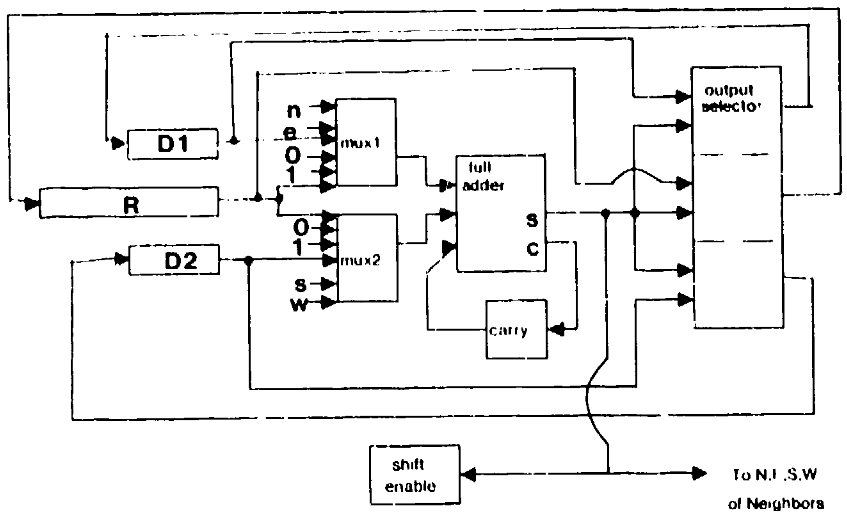


Figure 1. Processor data path.

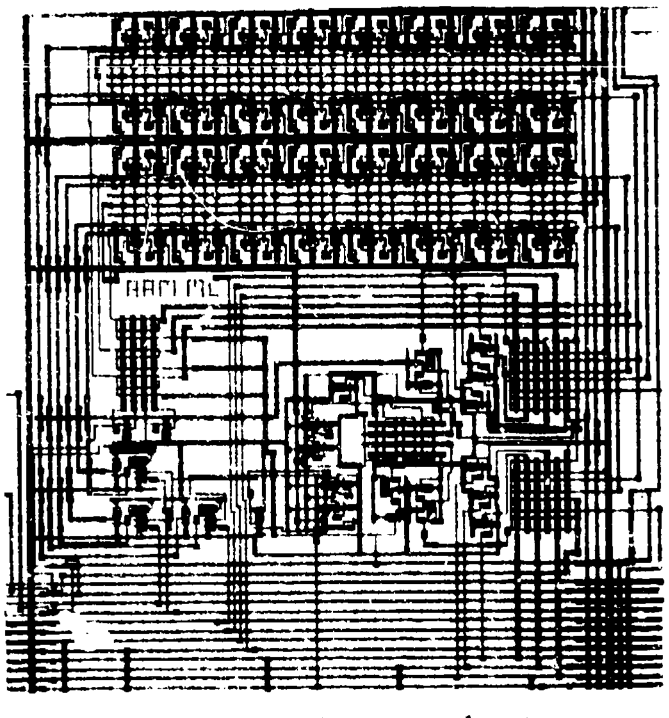


Figure 2. Actual processor layout.

processor connection scheme. We are currently considering either having each processor connect to all eight of its neighbors or having the processors arranged in a triangular array with each processor connected to six of its neighbors. By studying the fault tolerance of each of these topologies and using the yield data from the tests described above we will design processor interconnections to give us reasonable yields on the wafer-sized circuits. Testing the processors also affects the interconnection scheme since it must be done before any interconnections are changed, but the testing itself will rely on the interconnections to pass data on and off the wafer. We are also considering having buses along rows and columns of the wafer through which processors can communicate with each other and the outside world. These will simplify addressing a single processor for testing purposes, and will also allow rapid communication between distant processors.

Once we have decided on the topology of the interconnections, we will need to implement these interconnections. We are currently considering three different types of interconnections. "Soft" interconnections can be made by adding one more select input to MUX1 and MUX2 and controlling these inputs from a two-bit register. By making certain adder input selections not depend on the registers, these selection registers can be set to a known state after power is applied to the wafer. Testing can be done on all processors and the results of the testing can then be used to load the selection registers. "Firm" interconnections can be made using the same technology used in EPROMs, where interconnections can be enabled by storing a charge in the oxide layer over the base of a pass transistor. This charge can be removed by exposing the wafer surface to ultraviolet light, allowing occasional reconfigurations of the chip. "Hard" interconnections can be made using thin fusible metal links (the same technology used in PROMs). We are also investigating laser technologies for making and breaking links on wafer surfaces as a final processing step [7].

Performance Estimate

The performance improvements our system will provide are impressive. A Digital Equipment KL-10, which is a 6000-chip ECL design, can do a 36-bit addition in 520 ns if both operands are in the cache [8]. If we define an "operation" to be an 8-bit addition, the KL-10 can do 8.7 million operations per second. Our preliminary wafer will have 16,384 processors on it, so each processor will need to do 530 operations per second to match the KL-10. This represents a clock rate of 4.2 KHz, since each 8-bit addition requires eight clock cycles. Although timing tests are still in a very preliminary stage, we expect to be able to clock our device internally at close to 10 MHz, resulting in a performance improvement of at least 2500. The actual figure will probably be much larger since our system will avoid the overhead of computing addresses.

Conclusion

By making an image-sized two-dimensional array processor on a single silicon wafer, we are building a research tool for developing more powerful vision algorithms than current computing facilities allow. The system is particularly well-suited for algorithms involving local operations, and can be used to implement a large variety of algorithms using a general purpose processing element. We are using a fault-tolerant processor interconnection scheme to achieve high enough yield for this full-wafer circuit.

References

[1] Philip Marks, Low-level vision using an array processor, *Computer Graphics Image Processing* 14, 1980.

[2] M. J. B. Duff and D. M. Watson, The cellular logic array image processor, *Computer J.* 20, no. 1, 1977.

[3] David H. Hubel and Torsten N. Wiesel, Brain mechanisms of vision, *Scientific American* 241, no. 3, 1979.

[4] S. D. Fouse, G. R. Nudd, and V. S. Wong, Application of LSI and VLSI to image understanding, *Proc. ARPA Image Understanding Workshop*, April 1980.

[5] Robert W. Floyd and Louis Steinberg, An adaptive algorithm for spatial greyscale, *Proc. of the Society for Information Display* 17, no. 2, 1976.

[6] Tsuneo Mano *et. al.*, A fault-tolerant 256K RAM fabricated with molybdenum-polysilicon technology, *IEEE J. Solid-state Circuits* SC-15, no. 5, 1980.

[7] J. Randal Moulic and Walter J. Kleinfelder, Direct IC pattern generation by laser writing, *Proc. 1980 IEEE International Solid-state Circuits Conference*, 1980.

[8] DECsystem-10 Hardware Reference Manual. Digital Equipment Corporation, Maynard, Massachusetts, 1976. Also, timing information taken from on-site timing tests on the Stanford Artificial Intelligence Laboratory KL-10.

A RESIDUE-BASED IMAGE PROCESSOR FOR VLSI IMPLEMENTATION

S.D. Fouse, G.R. Nudd, and G.M. Thorne-Booth

Hughes Research Laboratories, 3011 Malibu Canyon Road, Malibu, CA 90265 and

P.A. Nygaard and F.D. Gichard

Carlsbad Research Center, 6155 El Camino Real, Carlsbad, CA 92008

ABSTRACT

This paper describes recent work undertaken at Hughes Research Laboratories, Malibu, California, in support of the DARPA Image Understanding (IU) program. The principal goal of the work is to investigate the application of VLSI technologies to IU systems and identify processor candidates well suited to VLSI implementation. One candidate that is very well suited to the VLSI technology is a programmable local-area processor with residue arithmetic based computations. The design and development of this processor, which operates on 5x5 kernel, are described. Of significant interest is an LSI custom circuit that we are developing and which will perform the bulk of the residue computations. In addition, an interface that will permit this processor to be controlled by a general-purpose host computer (e.g., PDP 11/34) is described.

1. INTRODUCTION

Our previous work (1,2) in developing image understanding architectures has concentrated on the analysis of the processing functions required for special-purpose LSI primitives. We have developed about 16 fixed and programmable primitives for real-time operation.

The work described here represents a significant shift in emphasis and an increase in capability. First, we have undertaken a detailed design and analysis of a number of complex processing operations, including line-finding (3) and texture analysis (4). This work has been carried out specifically with LSI and VLSI implementation in mind. Hence issues such as chip and function partitioning, data flow, local storage, and word-length are specifically emphasized. The results of the systems analysis and design for these operations are included in Section 2. From this work we have been able to configure a fully integrated real-time processor for each.

Of equal importance, and perhaps greater impact to military systems and robotics, we have configured, designed, and started to fabricate a VLSI processor that can form the basis of a fully programmable image understanding system compatible with commercially available host machines. The architecture itself uses residue arithmetic (5) to provide a highly regular and extendable structure. These issues are of great importance in the emerging VLSI era where design time and the ability to amortize the fabrication cost of many processors are essential elements. The VLSI processor now under development is configured on a single board with multiple copies of a single custom-built nMOS chip. Our estimates indicate that the processor will perform between 50% and 75% of the operations for line finding and texture classification. The modular nature of the machine can provide essentially variable precision as discussed in Section 3. The single custom-built chip has a complexity equivalent to approximately 6,500 transistors. However, with decrease in design rules from the present 5 μm to submicron we can anticipate building a single chip with some 80,000 transistors and design the full system around four identical chips.

A significant advantage of our approach is the compatibility with general purpose host machines, such as the DEC series, which are widely used in image analysis and understanding. We have therefore spent considerable effort in developing a UNIBUS interface so that the machine can be accessed through the host software. With the addition of the local area logic processor, to be developed in the next phase, we expect to demonstrate a fully programmable real-time processor.

2. SYSTEMS ANALYSIS AND DESIGN

The effective explortation of VLSI technology in image understanding systems requires that the processors developed be used in as wide a range of systems as possible. This requires that a wide variety of systems be analyzed for the purpose of determining commonality. Our approach has been to select three representative systems to analyze: a line finder, a texture analyzer, and a segmenter (6). Each system was studied and then a directed graph depicting the data flow was produced (7). The directed graph had nodes that were functionally complex, so the next step was to perform a logic design for the systems to determine the complexity of the nodes and of the systems. The logic design was done for the line-finder and the texture analyzer systems. For details of each design, see USCIP Report 990 (8). A brief summary of the results for each system are presented below.

The line-finder system decomposes into four major functions: edge detection, edge thinning, edge linking, and edge tracking.

A design was generated for each of these functions and Table 1 presents the number of gates each required. Similarly, the texture analysis system decomposed into five major functions: small-window convolution (5x5), small-window statistical calculation, scaling, large-window statistical calculation, and linear transformation. Table 2 presents the gate count for each system.

TABLE 1. GATE COUNT FOR LINE FINDER SYSTEM

EDGE DETECTION	178K
THINNING	190 GATES
EDGE LINKING	500
EDGE TRACING	12 MBIT MEMORY +5K LOGIC GATES

TABLE 2. GATE COUNT FOR TEXTURE CLASSIFICATION SYSTEM

5 LINE KERNEL GENERATION	15K GATES
5 x 5 CONVOLUTION	27K GATES/ CHANNEL
5 x 5 VARIANCE	10K GATES
NORMALIZATION	1K/CHANNEL
LARGE WINDOW STATISTICAL CALCULATION	8.25K/CHANNEL
TRANSFORM (M INPUT CHANNELS)	2.1K·M/OUTPUT CHANNEL

From the results of the directed graph analysis and the logic design, it is obvious that the function common to all three systems and the most complex, when measured by the number of gates, is the small window (5x5) convolution. This supports our decision to build a programmable 5x5 local-area processor as the basic VLSI module.

3. A RESIDUE-BASED IMAGE PROCESSOR

The work described in Section 2 motivated the design of a low-level processor that could perform the computationally intensive low-level operations for each of the three systems investigated. In addition to fulfilling the requirements of the three systems, we also wanted to select an architecture that could be extended to take advantage of the VLSI design and processing capabilities that are currently being developed. The architecture we selected is based on the technique of residue arithmetic.

3.1 Processor Description

We implemented our local area processor in residue arithmetic to take advantage of modularity, and hence ease of design, within the VLSI chip and extendability to handle arbitrary dynamic range and accuracy. The technique relies on the conversion, prior to computation, of all the data to relatively prime bases (we chose 31, 29, 23, and 19) and the subsequent decoding of the processed data back to binary numbers. If this overhead is accepted then the arithmetic itself is reduced both in complexity and in required dynamic range. This enables us to use look up tables, which in our case are programmable RAM, to perform the necessary arithmetic. Regularity, ease of VLSI design, and function density on the chip are significant advantages. Thus this approach is ideal for VLSI implementation.

A block diagram of a general residue processor is shown in Figure 1. Some of the advantages (e.g., modularity) and disadvantages (encoding, etc.) of this technique are clearly visible in this representation. The encoding and decoding, when compared to a binary processor, are overhead functions and can be the major disadvantage of a residue processor. However, this overhead cost can be reduced if enough computations can be performed while in the residue representation, and hence the encoding and decoding can be amortized over a large computation base. The clear advantage of this type of processor is its natural parallelism. Each parallel computation channel is independent, requiring no communication with its neighbors until the conversion from the residue representation to a binary representation is performed.

3.1.1 Kernel Generation and Encoding

Typically, the input to an image processor is a string of eight-bit data values generated by a raster scan of the image. Therefore, we must include in the processor the means for generating the two-dimensional kernel. This kernel generation function is most easily accomplished using a series of shift registers. For a five-line kernel, four shift registers, each one containing as many elements as there are pixels in a line, are required to generate five adjacent lines of video. For our particular application the shift registers are 8 bits wide and 512 elements long.

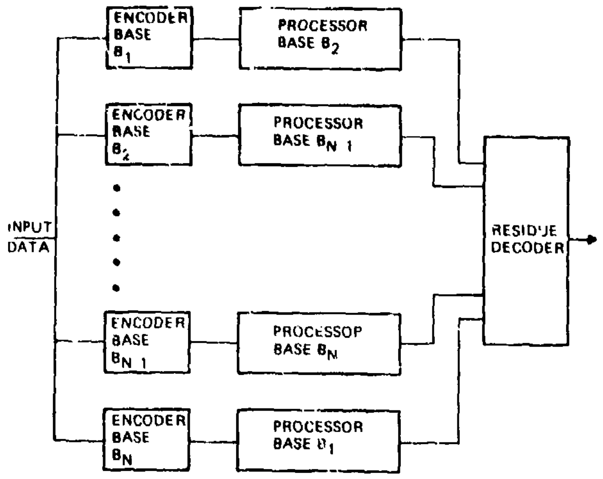


Figure 1. General Structure For Residue Processor

Before the input data can be processed by a residue processor it must be converted from a binary representation to a residue representation. This conversion requires that we calculate

$$(X \bmod B_1,\ X \bmod B_2,\ \ldots,\ X \bmod B_n),$$

where X is the value of the input data and B_i is the ith base. For our case, since we operate on a 5x5 kernel, we must perform this calculation on the input for each of five lines of video and for each of four processors (equivalent ot the four bases). The simplest way to perform this calculation for a general set of bases is to use read only memories (ROMs). By connecting the input data to the address lines of the ROM and looking at the data lines of the ROM for the output, a look-up function is performed. For our particular processor, which will support an eight-bit input dynamic range and bases that can be encoded in five bits or less, the size of an encoding ROM is 256x5 bits. Figure 2 shows the block diagram for the kernel generation and encoding portion of a four-base five-line processor. Each of the five ROMs for each base are programmed identically.

An alternative way to perform these two functions would be to encode before the kernel is generated. The major drawback of this technique is that the memory requirements are much greater for the kernel generation process. For the system we are currently constructing, each line delay would need to be 20 bits wide as opposed to eight bits wide for the method we chose.

3.1.2 A Programmable Residue Computation LSI Circuit

The actual computations on the image data will be performed by a custom LSI circuit which is currently being processed at the Hughes Carlsbad Research Center. The circuit will process a

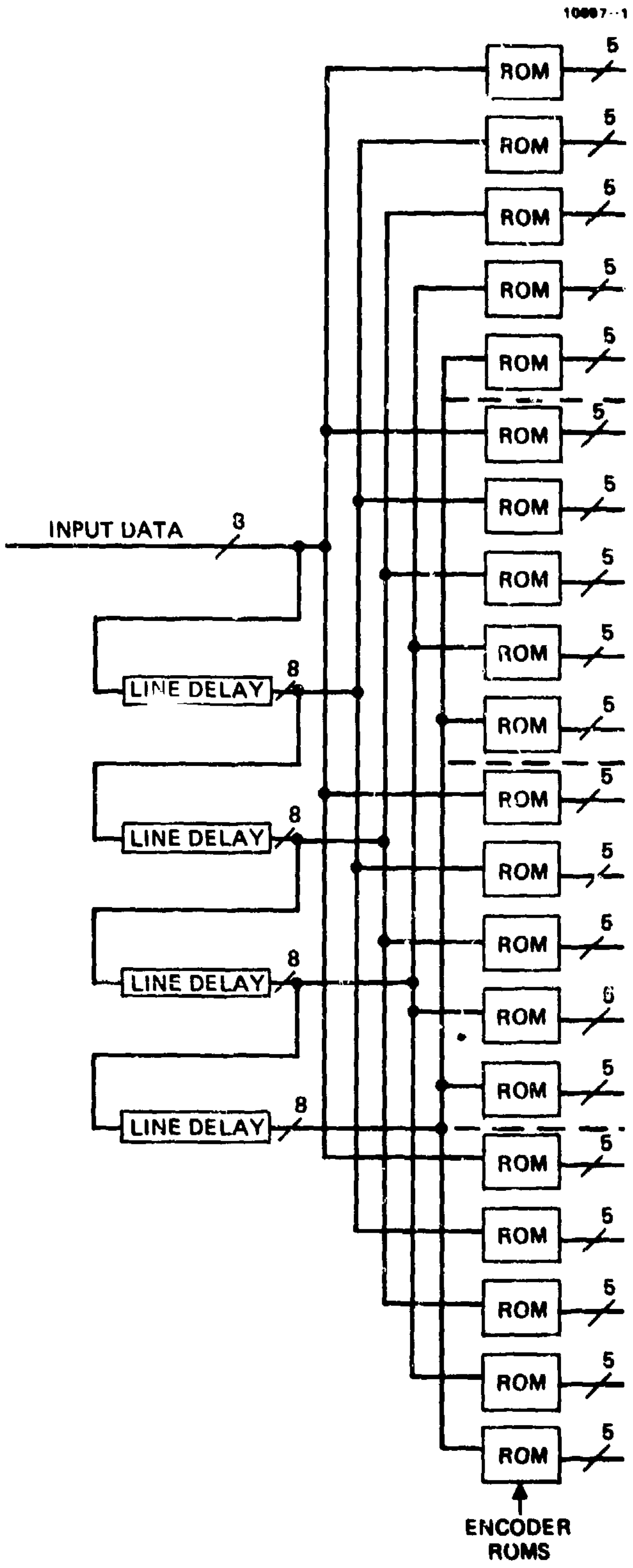


Figure 2. Kernel Generator Encoding for 5x5 Processor

5x1 kernel and is capable of performing computations of the form

$$y = \sum_{i=1}^{5} f_i(X_i) ,$$

where y is the output value, X_i is the five elements in the kernel, and f_i represents polynomial functions of a single variable.

A functional block diagram of the circuit is shown in Figure 3. The word size for this is five bits, which limits the prime bases used to a value of 32 or less. The circuit is designed to accept a five-bit input word which is clocked into a five-element shift register. The contents of each register element is then shifted to the next register. The five-bit data in each of the shift register elements is used to address a look-up table, which is a 32x5 random access memory (RAM). This look-up operation performs a unary operation such as a multiplication by a constant or a squaring operation. The outputs of the five RAMs are then summed modularly to produce a five-bit output, the base of the modular addition being programmable by external control of the circuit. Since the look-up tables that perform the unary operation are composed of RAMs, the circuit can be programmed for many different computations, such as different weights for a convolution or different powers of a number for a statistical calculation.

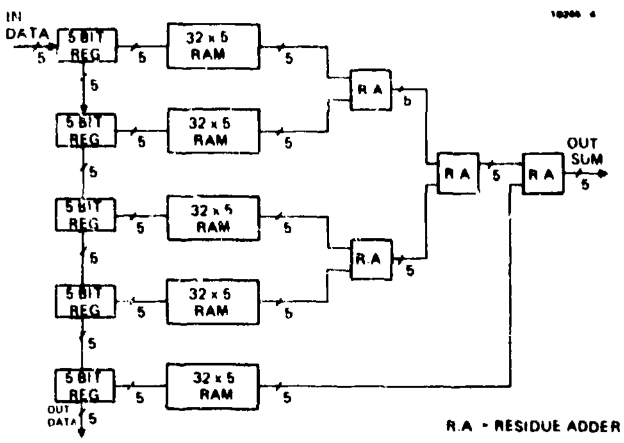


Figure 3. A Functional Block Diagram for 5x1 Residue Processor Circuit

A detailed schematic of the circuit is shown in in Figure 4. In addition to having five bits of input data and five bits of output data, an additional set of data lines is included in this design. These data lines, which are bi-directional, serve a multipurpose role for control and testing. When used as input data lines they can be used to program the base of the modular addition and to program any of the five look-up tables. When used as output data lines they can read the look-up tables to verify the operation of the circuit.

This circuit is being fabricated using the nMOS technology and has been designed to accept a 10 MHz data rate. To achieve this data rate, pipeline techniques were used, and the resulting latency for this circuit is seven clock cycles. The circuit will be packaged in a 28-pin dual in-line package. Figure 5 is a photograph of the layout of the chip, which will be available for testing in April 1981.

To utilize this circuit (with a 5x1 kernel) in a 5x5 local area processor, multiple copies of the circuit need to be used as well as additional logic to combine the outputs of the individual circuits. For each base, five of these circuits are used, one for each line of the kernel. In addition, four 1,024x5-bit ROMs are used to sum the outputs of the five circuits. ROMs are used instead of adders because the additions must be done modularly. Figure 6 shows the block diagram of the processor, including the encoding and computation portions.

3.1.3 Decoding

The last portion of the processor is concerned with the conversion from the residue representation to a binary representation. This conversion could certainly be done the same way as the encoding, by table look-up, but there is a severe problem with that approach. For our particular system, to convert four 5-bit values, the decoder would require a memory 1 million elements wide with each element being 17 bits deep. This table is certainly attainable but the approach is not extendable. If an extra base is required, so that five 5-bit values need to be converted, the memory requirements increase to 33 million elements, each being greater then 20 bits deep.

There are two conversion methods that do not require these large memories. One is based on the Chinese remainder theorem and the other is based on a mixed radix representation. (For a complete discussion, refer to Ref. 9.) This paper will focus only on the particular implementations of these techniques and the rationale for selecting one over the other.

To be able to reasonably discuss either of the two conversion methods some notation must be introduced. If B is the base vector whose elements are the bases used for the computations,

$$B = (b_1, b_2, \ldots, b_k),$$

R is a scalar whose value is equal to the dynamic range of the processor, which is given by

$$R = \prod_{i=1}^{4} b_i$$

and X is the value we wish to encode into the residue representation, then RX, the vector whose elements are the data values for each of the computation channels, is given by

$$RX = (rx_1, rx_2, \ldots, rx_k)$$

where

$$rx_i = X \text{ MOD } b_i, \; i=1 \text{ to } k.$$

The Chinese remainder conversion process is based on the following property. If

$$kX \rightarrow (rx_1, rx_2, rx_3, rx_4)$$

then

$$RX = [(rx_1,rx_2,0,0) + (0,0,rx_3,rx_4)] \text{ MOD } R.$$

Figure 7 shows a system which performs this conversion and which requires only two blocks of memory 1,024 elements wide and two adders. The adders need only be as large as the accuracy required of the system. Typically, for image processing systems, the output dynamic range and the input dynamic range are equal and thus the adder complexity can be relatively small.

The second conversion scheme considered is based on the mixed radix method, but is simplified by the fact that the output dynamic range can be approximately eight bits. The method can be explained by imagining an iterative process where at every iteration the smallest base is eliminated by dividing the value by that base. Dividing essentially reduces the dynamic range of the value and thus eliminates the need for the extra base. Of course, since we are limited to a strictly integer system, we must make sure that the value is evenly divisible by the smallest base. This can be done by rounding up or down so that the element in the residue vector for that base is zero. Figure 8 shows an architecture for a four-base system that

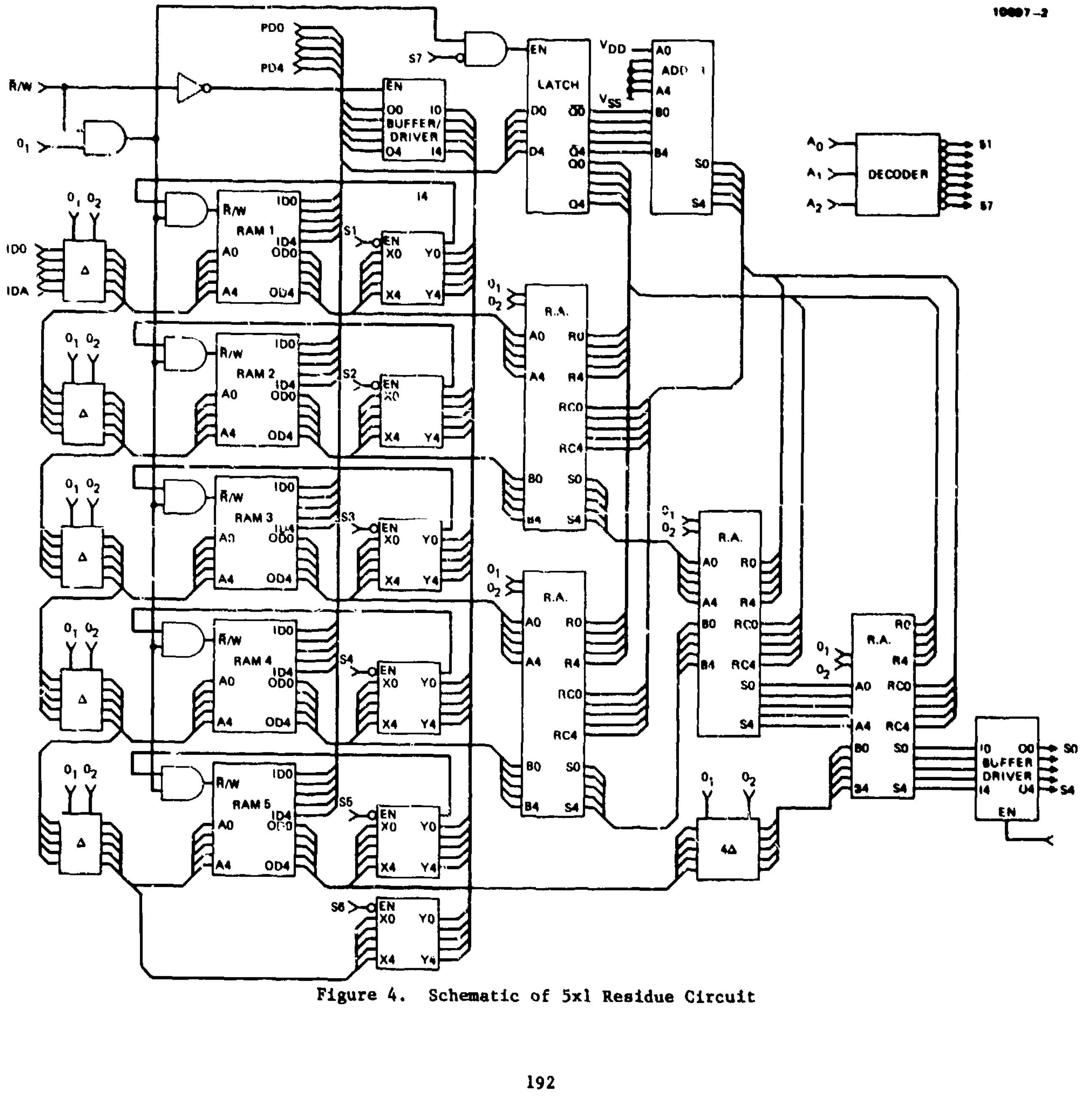


Figure 4. Schematic of 5x1 Residue Circuit

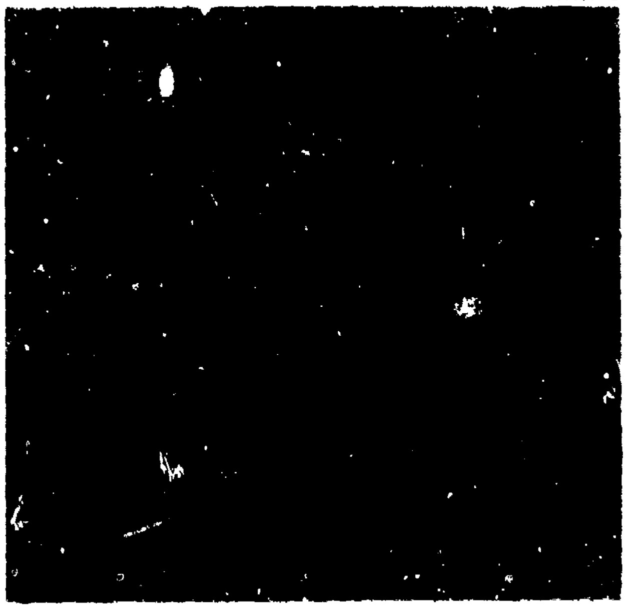


Figure 5. Photograph of CRC 181 Layout

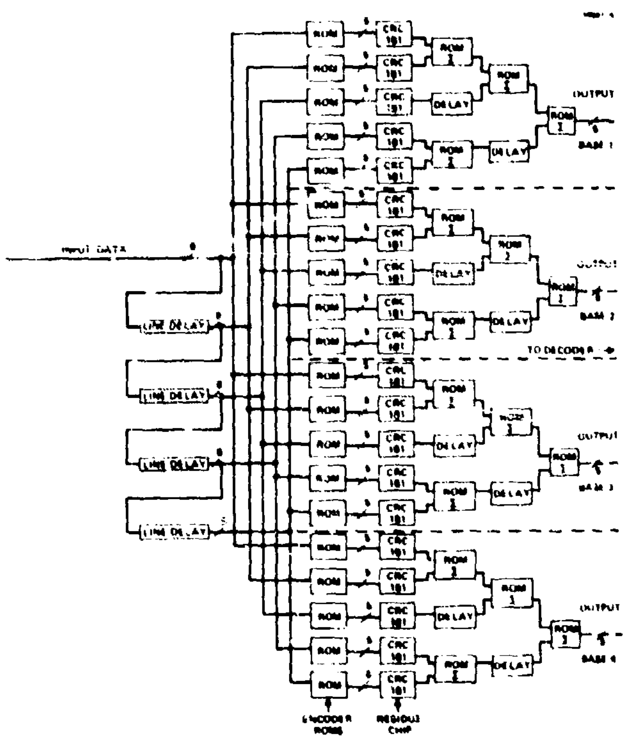


Figure 6. Structure of 5x5 Processor Utilizing 5x1 Processor Circuits

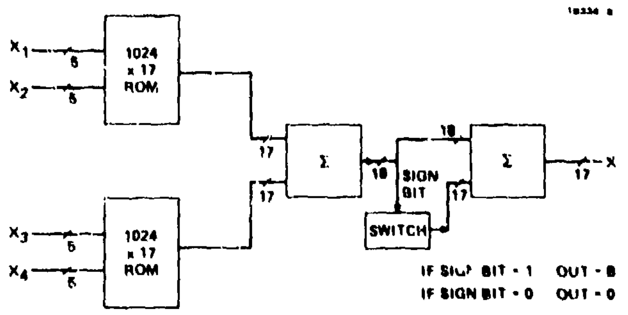


Figure 7. Chinese Remainder Theorem Residue Decoder (4 Base System, 5 Bits/Base)

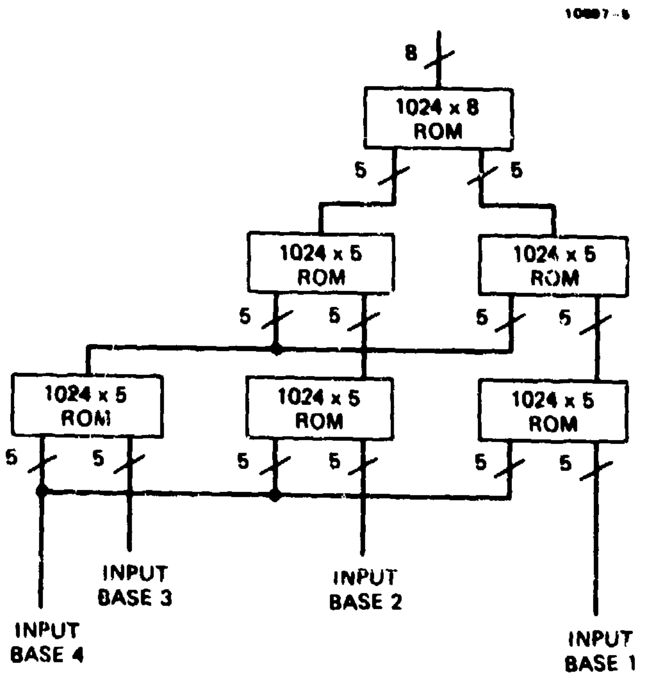


Figure 8. Mixed Radix Based Residue Decoder (4 Base System, 5 Bits/Base)

performs this mixed radix like conversion. At the bottom level of this tree structure the fourth base is eliminated. At the next highest level of the tree the third base is eliminated. Finally we are left with two base values that can be decoded with a simple look-up table. This system has been simulated, and the computer programs exist that can generate the contents of the ROMs for this conversion process for an arbitrary set of bases.

We chose the mixed-radix-based conversion process to be implemented for our processor for two reasons. First, the method does not require any logic other than ROMs. This tends to make it more flexible and reliable. Second, the method appears to be easily extended to more bases, by simply extending the decoding tree. Extending the Chinese-remainder-based process would require either larger ROMs or more adders, either way being less attractive way than the mixed-radix-type conversion.

3.1.4 Programming and Control

A major problem in the fabrication of this processor is gaining access to each of the 20 custom residue chips for the purpose of programming the look-up tables. Each chip has three address lines to select one of five RAM structures, a read/write line, the five bi-directional programming data lines, and a control line for the data line drivers. These 10 control lines must be brought out for each of the 20 custom chips for a total of 200 control lines for the purpose of programming. However, by bussing lines where possible and by using a peripheral interface chip, the Intel 8255, the number of lines that are actually brought out of the processor is reduced to 16.

Figure 9 shows the structure that will be used to program the processor. The three address lines and the bus driver control lines are brought from each custom chip to an 8255. One 8255 is able to control five of the custom chips, since the 8255 has 24 lines available through three 8-bit ports. Thus four 8255s are required to control all 20 of the custom chips. In addition, the five program data lines and the read/write line are bussed between each of the chips and these six lines are brought to a fifth 8255. The eight input data lines of the 8255s are bussed as well as the two-bit port select address lines. Finally, we bring out each of the 5 chip select lines to a binary decoder, allowing selection of a single 8255 using three control lines.

To use this structure to program a given RAM element in the processor requires the following steps. Initially, the fifth 8255 is selected and the data to be programmed are written to the port containing the five program data lines. Next, the 8255 that controls the chip that the desired RAM element is on is selected, and the code to select the desired RAM structure is written to the proper port. Next, the address of the desired RAM element is provided on the input data line of the processor, and then the address is shifted so that it is addressing the proper RAM structure. Finally, the write data line is strobed to complete the programming sequence. This sequence can be accomplished by three 16 bit data transfers. For a processor using 31, 29, 23, and 19 as the modular bases, a total of 2,550 RAM elements need to be programmed. Thus, if three word transfers are required for each RAM element, a total of 7,650 word transfers are required to completely program the processor.

The processor that involves all of the functions described above is currently being fabricated. The majority of the electronics will be on a single wirewrap board, but the line delays will be in a separate box. A picture illustrating the progress on the wiring of the board is shown in Figure 10.

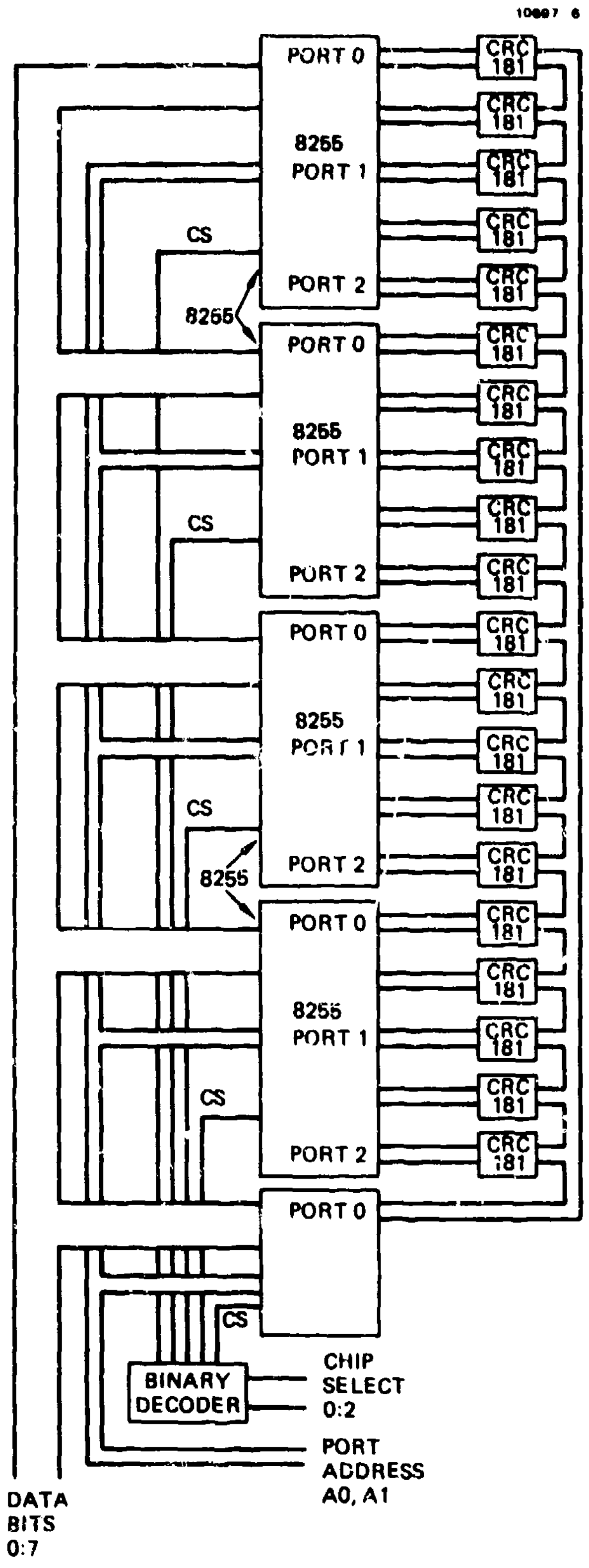


Figure 9. Structure for Programming and Control of Residue Processor

Figure 10. Photograph of Processor Wirewrap Board

3.1.5 A VLSI Version of a Residue Computation Chip

Even though the custom chip is performing the bulk of the computations, some extra circuitry is still required. Most of the extra circuitry is necessary because we are using a custom circuit based on a 5x1 kernel. The next step then, once this processor is tested and demonstrated, is to develop a 5x5 residue custom circuit and to fabricate a processor that would utilize these VLSI circuits.

Ideally, the 5x5 circuit should include the circuitry for generating the five-line kernel. This would mean that there would be five bits for input and five bits for output. If the kernel generation circuitry is performed off of the chip, 25 lines for input would be required, or at least a high-speed multiplexer would need to be on the chip. On the other hand, if a simple line delay is used to generate the five-line kernel, the circuit would be too inflexible, since it may not be suited to certain applications. This can be avoided by augmenting the shift register with logic to control the clocking. By multiple clocking the shift registers can be made to delay any length up to the maximum length. In other words, we would be constructing an elastic delay line.

Figure 11 shows a block diagram for the data flow of a 5x5 residue circuit. This circuit includes four delay lines, probably 512 elements long, 25 registers for generating the 5x5 window, 25 RAM structures, 24 modular adders, and some delay stages. In addition, programming, control, and testing of this circuit must be considered. A great deal about these issues can be learned by using the processor currently being fabricated.

Both the current chip, the 5x1, and the next chip, the 5x5, have been sized to get a quantitative measure of their complexity. The CRC 181 has a device count of approximately 6,500, of which the RAM portions of the circuit take up 4,500 devices. For the 5x5 custom circuit the device count will increase to 80,000. One of the reasons for the high device count is the addition of the line delays, which account for 50,000 devices (for static memory cells). The total number of devices for random logic is about 7,000, which is low considering that the circuit will have a throughput of 500 million operations per second.

As stated, going to a 5x5 circuit will greatly reduce the extra circuitry required to construct a 5x5 processor. Figure 12 shows a block diagram of a system utilizing a 5x5 circuit. With the VLSI circuit the package count for the data flow portion of the processor will be only 14. This is compared to a package count in excess of one hundred for the current design. The power and size will be greatly reduced, thereby permitting the processor and the DEC UNIBUS interface to be put on a single card.

3.1.6 Functional Capabilities

Although the primary motivation of developing this processor was that the systems we investigated required 5x5 convolutions, the processor is capable of performing a wider range of computations. The reason for this flexibility is that we used a look-up table to perform a unary operation, and the table is completely programmable. The general form of the computation that can be performed by the processor is

$$y = \sum_{i=1}^{25} f_i(x_i) \quad ,$$

where y is the output, the x_i represents the 25 elements in the 5x5 kernel, and f_i represents polynomial functions of a single variable. Each f_i is completely arbitrary and need not have any relation to the other f_i. By selecting subsets of the f_i to be identical to zero, we can program the processor to perform point transforms, one-dimensional transforms of any size up to 5x1, and two-dimensional transforms of any size up to 5x5. Table 3 lists some of the functions that can be performed by this processor.

4. UNIBUS INTERFACE

The processor, as mentioned before, is designed to accept data at 100 nsec intervals. The reason for this high-speed design is to allow real-time stand-alone operation. This means, however, that when the processor is used as a peripheral device attached to a general purpose computer, the data

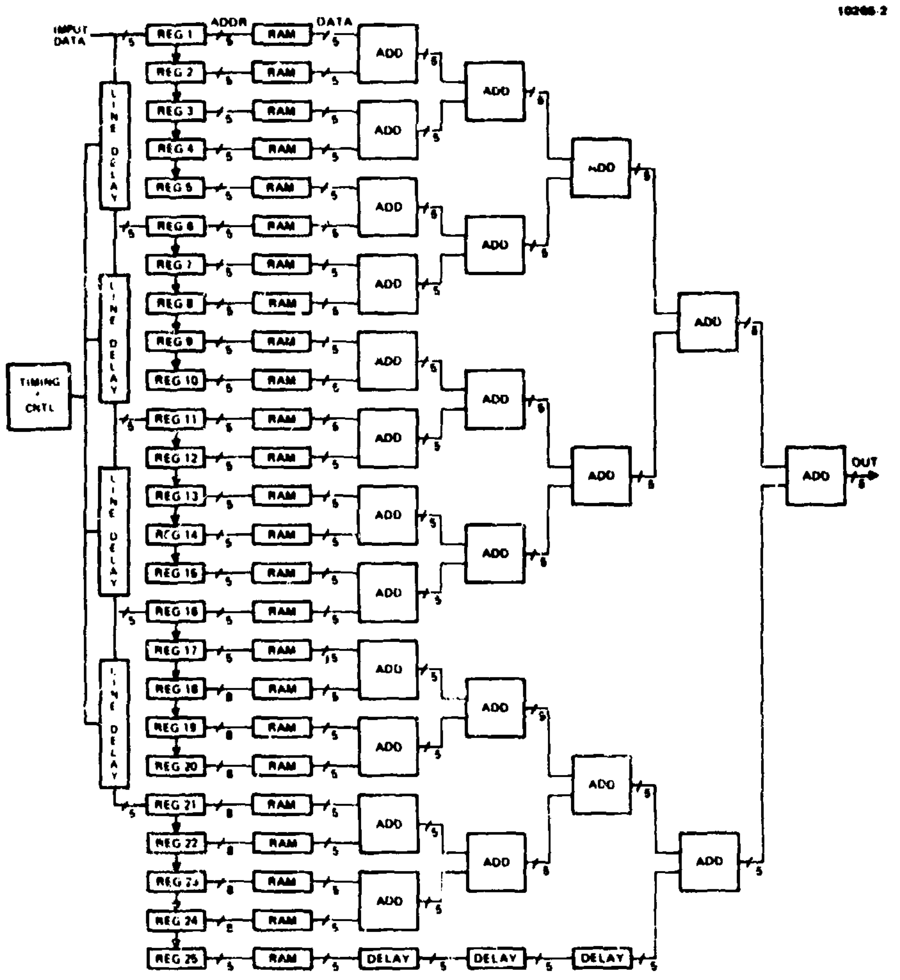


Figure 11. Data Flow for 5x5 Residue Custom Circuit

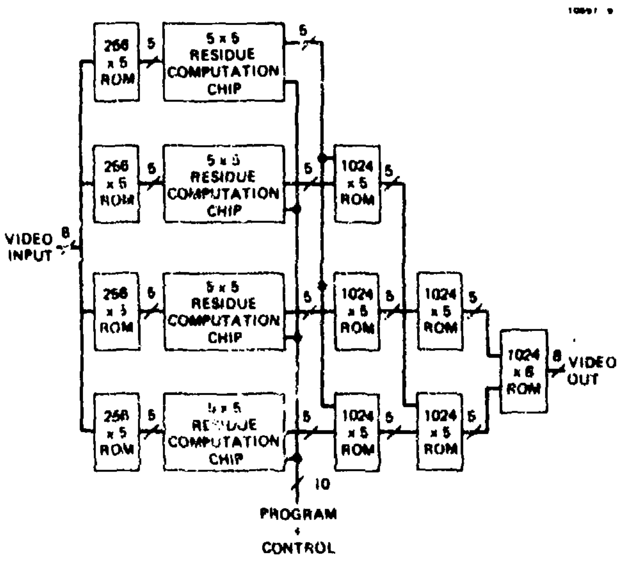


Figure 12. Residue Processor Based on 5x5 Custom Chip

TABLE 3. FUNCTIONAL CAPABILITIES OF RADIUS

10897-14

POINT OPERATIONS POLYNOMINAL FUNCTIONS CONTRAST ENHANCEMENT
1 - DIMENSIONAL OPERATIONS INTEGER COEFFICIENT TRANSFORMS POLYNOMINAL FUNCTIONS
2 - DIMENSIONAL OPERATIONS EDGE ENHANCEMENT STATISTICAL DIFFERENCING LOW PASS/HIGH PASS FILTERING SHAPE MOMENT CALCULATIONS STATISTICAL MOMENT CALCULATIONS INTEGER COEFFICIENT TRANSFORMS TEXTURE ANALYSIS

transfer will be limited by the memory cycle of the general-purpose computer and not by the processor speed. This means, that to get optimal use of the processor, we need the fastest type of transfer available between the processor and the main memory, where the data to be processed will reside. The direct memory access (DMA) type of transfer is the fastest type of transfer that a general-purpose computer can support, since it does not require processor intervention. For DEC UNIBUS applications, the fastest data rate one could expect is approximately 1 MHz.

The type of interface we design should then be able to provide a DMA transfer capability for both the programming data and the image data. For either type of transfer, it is essential that the interface be controlled to select the memory location from which the data are to be transferred and to select the number of words to transfer. For program data transfers, the interface will only be required to transfer one way at any time. The transfer will be to the processor while in a program mode and from the processor while in a test mode. For image data transfers the interface must be able to transfer data both ways, for input and output. The simplest alternative to handle this bi-directional transfer of data (from a hardware point of view) is to transfer the output data to the same memory location the input data came from, i.e., write the output image over the input image.

DEC devices exist that can provide the DMA transfer capability as well as provide several control lines to the peripheral device to allow multiple transfer modes. One such device is the DEC DR11B UNIBUS parallel interface. Our plan is to use this device to provide the DMA capability and to design a custom interface to permit the specific transfer modes. The arrangement suggested is shown in Figure 13.

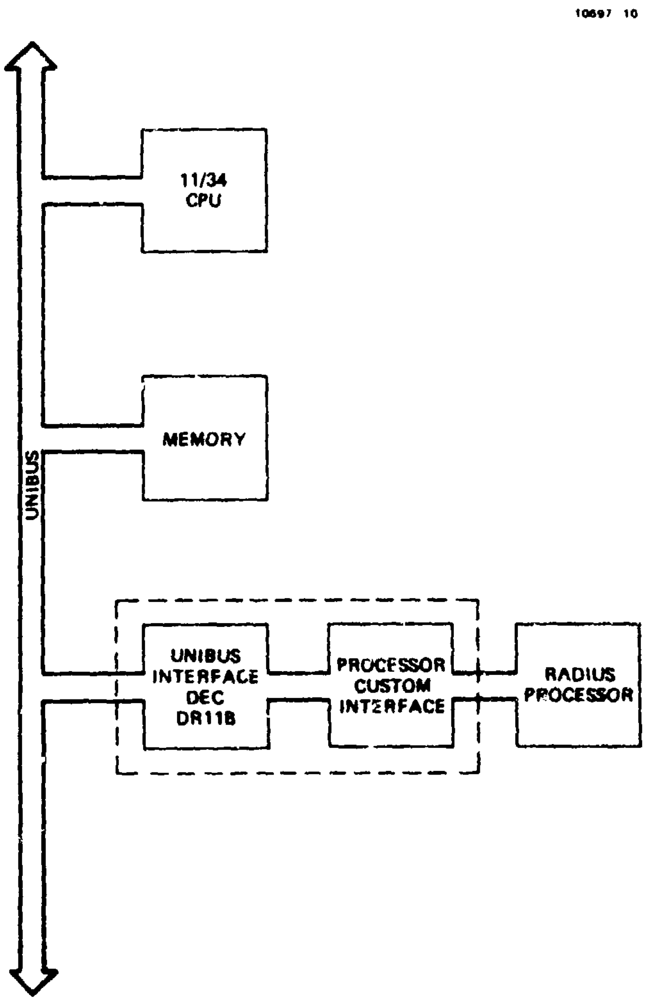


Figure 13. Commercial/Custom UNIBUS-Processor Interface

The custom interface will need to interpret the control lines from the DR11B and decide if the transfer is for program data or image data. If it is program data, the interface will simply pass the data to the 16 program data lines. If it is an image data transfer, then the custom interface is more complex. Since it is a 16-bit transfer, the data will contain two pixels. So following the transfer, the interface must first pass one byte to the input data lines and then the next byte. Simultaneously, the interface must load the first output image data into one byte of the 16-bit output data register and then the next output data into the other byte of that register. Finally, the output data register's contents are transferred to the DR11B which writes it to main memory. A preliminary schematic of a system that can perform these types of transfers is shown in Figure 14.

5. SUMMARY AND FUTURE WORK

We have described the work undertaken to design VLSI processors for these widely used systems: line-finding, texture classification, and segmentation. From this work, we believe we can, if required, build the necessary hardware. However, of greater impact, we have identified and started to build a fully software-programmable low-level processor for 5x5 operations. The circuitry described relies on a special-purpose VLSI chip with 6,500 components. Using this, and the interface designed to hook the processor to commercial general-purpose machines, most low-level arithmetic operations over a 5x5 kernel can be performed. This work will continue and the full system will be demonstrated using our in-house PDP 11/34. We then anticipate making this available to interested government researchers in this field.

Our future plans include the investigation and possible development of a single ultra-high-density circuit to include the full processor and the development of a compatible logic processor.

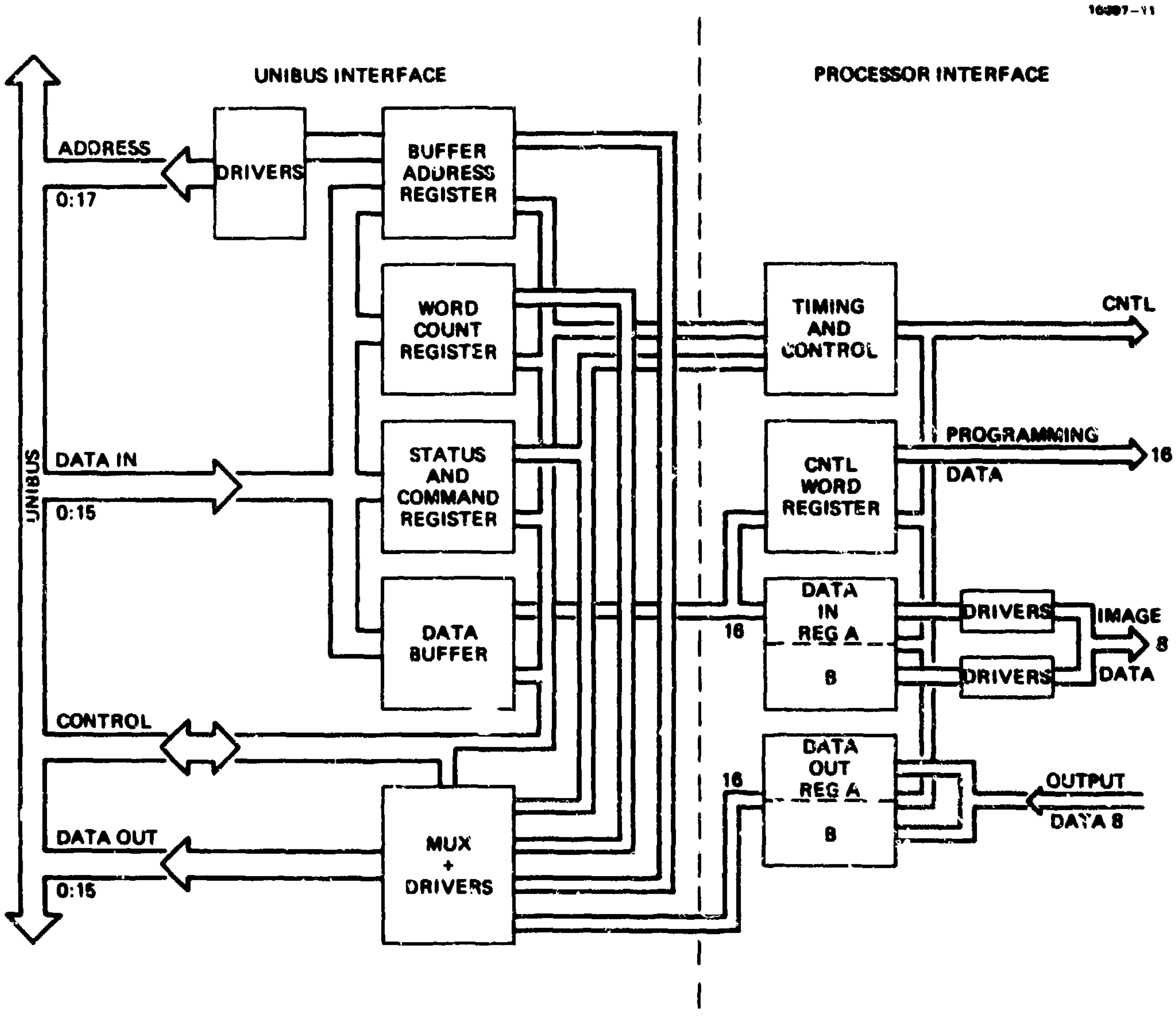


Figure 14. Bus Structure of UNIBUS-Processor Interface

REFERENCES

1. G. R. Nudd "Image Understanding Architectures," National Computer Conference, May 1980, Anaheim, California, AFIPS Conf. Proc., Vol 49, pp. 377-390.

2. S. D. Fouse, G. R. Nudd, and P. A. Nygaard, "Implementation of Image Pre-Processing Functions Using CCD LSI Circuits," Proc. Society Photo-Optical and Instrumentation Engr., Vol. 225, pp. 118-130, SPIE Conf., April 1980, Washington, D.C.

3. R. Nevatia and K. R. Babu, "An Edge Detection, Linking, and Line Finding Program," USC IPI Report No. 840, Sept. 1978.

4. K. I. Laws, Textured Image Segmentation, Ph.D. Thesis, USC, January 1980.

5. N. Szabo and R. Tanaka, Residue Arithmetic and its Applications to Computer Technology, (McGraw-Hill, New York, New York, 1967).

6. R. Ohlander, K. Price, and D. Raj Reddy, "Picture Segmentation using a Recursive Region Splitting Method," Computer Graphics and Image Processing (1978).

7. S. D. Fouse, G. R. Nudd, and V. S. Wong, "Application of LSI and VLSI to Image Understanding Architectures," Proceedings Image Understanding Workshop, April 1980.

8. S. D. Fouse, V. S. Wong, and G. R. Nudd, "Advanced Image Understanding Using LSI and VLSI," USCIPI Report 990, Sept. 1980, pp. 164-204.

9. A. Huang, Number Theoretic Processors: A Celluar Array Architecture, Ph.D. Thesis, Stanford, October 1980.

SECTION II

PROGRAM REVIEWS

BY

PRINCIPAL INVESTIGATORS

Image Understanding Research at CMU

Takeo Kanade and Raj Reddy

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Image Understanding Research at CMU mainly concerns three research areas: theory for understanding 3-dimensional shapes; integrated system demonstration for photo interpretation (database and interactive/automatic image interpretation techniques); and special devices and computer systems for image understanding. In this report we will present the CMU views and our recent representative progress for the first two areas.

Theory for Shape Understanding

Images convey the shape information in a very complicated manner. Many factors are interwoven into the imaging process: surface property, surface orientation, texture, class of objects, etc. They individually provide constraints on the shapes that the image depicts. Historically, most vision programs have used these constraints in a vague, unformalized manner.

At CMU we believe that it is an important challenge in image understanding to formulate theories of shape understanding from images: what constraints are provided by individual properties which are observable in the image, and how they can be aggregated into consistent shape interpretation. We have been actively investigating geometrical aspects of image constraints for extracting shape from static, monocular images:

- Shape recovery from line drawings [3] [4]
- Shape-from-Texture paradigm [5]
- Mapping image properties into shape constraints [2]

We will review our recent progress in the last two topics, together with the research on 3-D shape sensing and analysis.

Shape from Texture

Kender finished his Ph.D thesis on Shape-from-Texture [5]. Not only did the research produce many interesting results on texture analysis, but the Shape-from-Texture provides a computational paradigm. In the image forming process, surfaces are perspectively projected onto two-dimensional regions of the imaging retina. The images of the texture constituents which define a surface are distorted by local surface orientation, relative surface distance, and the characteristics of the imaging device. The task of the visual processor is to deconvolute these effects. Recovery of the scene characteristics depends on simplifying assumptions about the physical world. These are the notions of texture regularity, and of surface opacity and smoothness. The general paradigm exploits each of these assumptions in the creation and refinement of the analytic framework.

The fundamental conceptual and representational tool is the normalized textural property map (NTPM). Intuitively, this map relates a given two-dimensional image texel (texture element) to the small class of three-dimensional constituents which may have been its source in the scene. More precisely, it is a way of deprojecting the effects that surface orientation has on primitive textural properties such as slope in the image, length of major axis of elongation, etc. The map summarizes the answers to the question, "what would the textural property (e.g., real slope, real length, etc.) had to have been in the scene in order to observe the given textural property in the image?".

The NTPMs for more than one primitive texels are combined or intersected to derive a specific surface orientation. Here, additional heuristics about the physical world are invoked. Typically, they are the continuity of constituents (regularity), the continuity of local surface orientations (smoothness), and repetition of identical texel constituents (structural texture). Figure 1 is an example of recovering the surface orientation of a

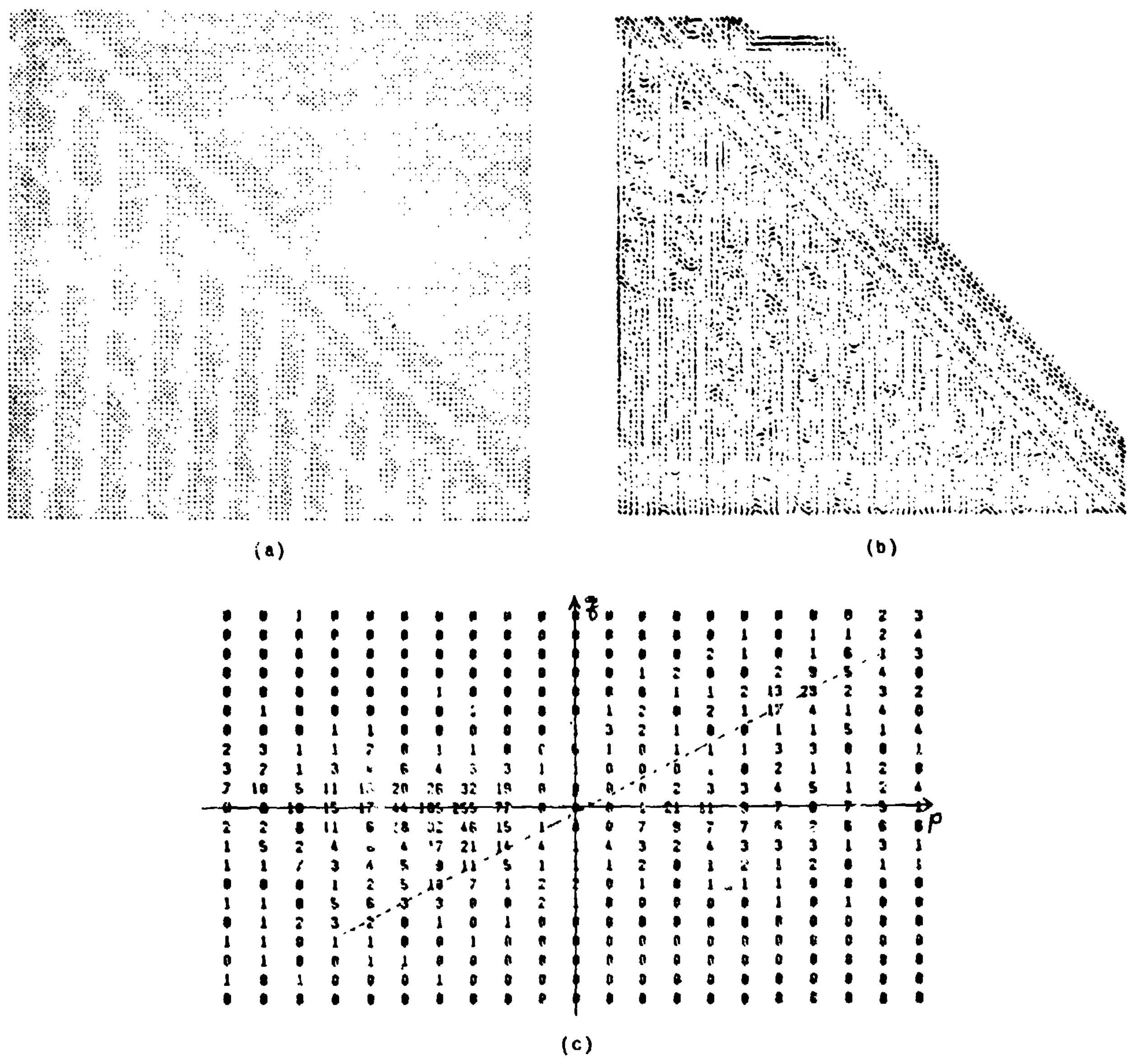


Figure 1: Recovery of a surface orientation of a building face

(a) A fragment of a digitized image of a building face. The surface orientation is approximately (p,q) = (0.4,0).
(b) The edge image.
(c) The gradient space accumulator array (portion near (p,q) = (0,0)). The surface orientation of the building face is given as the intersection of the two virtual lines: one horizontal, the other diagonal. They respectively stem from the vertical parallel lines and the diagonal parallel lines in the building face.

building face: it uses image slopes of texels and an assumption that near parallel image lines are parallel in the scene.

Following are some of the most illustrative results of Kender's thesis:

<u>*Slope in the Image Applied to Gravity*</u>

- Since most images are taken in an environment pervaded by a force that strongly orients objects with respect to it, certain heuristics regarding "up", "horizontal", and other gravity-based terms make image understanding simpler. For example, under perspective, assuming that a certain direction (usually, the y direction, through the center of the image) is vertical enables the orientation of the assumed ground plane to be determined by using a single near-vertical texel.

The Paradigm Applied to Length and Spacing

- The paradigm is applicable directly to image texels that have linear measure. Actual line elements, or virtual line elements (spacings) are analyzed in identical ways.
- The general problem of arbitrary line lengths is tractable in all cases under orthography, but only under special cases under perspective. The rotational coupling of the gradient space ensures that the orthographic NTPM has only one free parameter (the image length). However, the perspective NTPM has four: three for position, and one for scale.
- Under orthography, the assumption of equal length in the scene reduces to the case of the Kanade orientation hyperbola.
- Under perspective, the assumption of equal scene length plus the assumption of parallelism in the scene induces a simple graphic construction that directly gives the vanishing line, and therefore the surface orientation.
- Under perspective, the assumption of equal scene length plus the image phenomenon of colinearity create a one-dimensional, straight-line gradient space constraint.
- The colinear-equal assumption gives a unique vanishing point that can be determined exactly by using a Hough-like accumulator method that generates parabolas or hyperbolas in a transform space.

The Paradigm Applied to Area and Density

- The NTPM of density under orthography is identical to the reflectance map of a Lambertian surface. This gives strong theoretic support to the popular method of blurring textures into shades of grey for scene analysis purposes.
- Under perspective, the blurring of textures is hazardous, as it is crucially dependent on second order distribution statistics.

The Paradigm Applied to Intensity and Contour

- Texture and illumination share many parallels. Most often, the two phenomena (shading and pattern) are intermingled. Of the two, texture appears more "robust".
- Shading can also be analyzed under perspective. The simplest case of shading (the Lambertian sphere) has the same analysis under perspective as it does under orthography.
- The constraint curves generated by some shape-from-occluding-contour methods are identical to both those used in shape-from-shading and those used in shape-from-texture. All three have the same, simple graphic interpretation.

The Gaussian Sphere As the Preferred Representation

- The gradient space is deficient in that it is only half a space; the Gaussian sphere is a natural extension with a number of preferable properties [6]. However, the surface of a sphere is hard to handle in a straightforward way in a planar computer.
- The Gaussian sphere preserves many of the pleasing properties of the gradient space.
- Other representations may be occasionally useful: the inverted gradient space ((1/p,1/q) instead of (p,q)), for example. Especially intriguing is the problem of how to simplify several half-angle formulae.

Mapping image properties into shape constraints

Certain image properties, such as parallelisms, symmetries, and repeated patterns, provide cues for perceiving 3-D shape from a 2-D picture. Kanade and Kender [3] demonstrated how we can map those image properties into 3-D shape constraints by associating appropriate assumptions with them and by using appropriate computational and representational tools.

Skewed Symmetry

One representative example is the concept of *skewed symmetry*: a property of 2-D shapes in which the symmetry is found along lines not necessarily perpendicular to the axis of symmetry, but at a fixed angle to it. Formally, such shapes can be defined as *2-D affine transforms of real symmetries*. (Figures 2 (a)(b)(c) show a few examples.) There is a good body of psychological experiments [12] which suggests that human observers can perceive surface orientations from figures with this property. This is probably because such qualitative symmetry in the image is often due to real symmetry in the scene. Let us associate the following assumption with this image property: *"A skewed symmetry depicts a real symmetry viewed from some unknown view angle."*

We can map this assumption into the constraints on surface orientation. Let $G = (p,q)$ denote the gradient of the plane which includes the skewed symmetry, and α and β denote the 2-D directions of the skewed symmetry's axes. It can be shown that such a skewed symmetry in the picture can be a projection of a real symmetry if and only if the gradient is on the hyperbola shown in Figure 3, which is given by:

$$\cos(\alpha-\beta) + (p\cos\alpha + q\sin\alpha)(p\cos\beta + q\sin\beta) = 0$$

Kanade [4] has shown that this constraint plays important roles in perceiving quantitative shapes from line drawings of polyhedra. This *quantitative* shape recovery is an essential advance from Huffman-Clowes-Waltz type labeling, which simply could characterize shapes of polyhedra *qualitatively* (convex or concave edges, etc).

Affine-Transformable Patterns

A more general class of image properties that Kanade and Kender [2] have found is affine-Transformable patterns. In texture analysis we often consider small patterns (texel) by whose

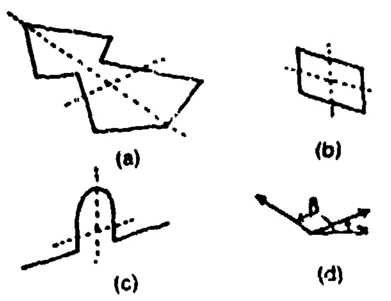


Figure 2: Skewed symmetry

A skewed symmetry defines two directions in the image skewed-symmetry axis (α) and skewed-symmetry transverse axis (β) The skewed-symmetry assumption assumes that they are perpendicular in the scene.

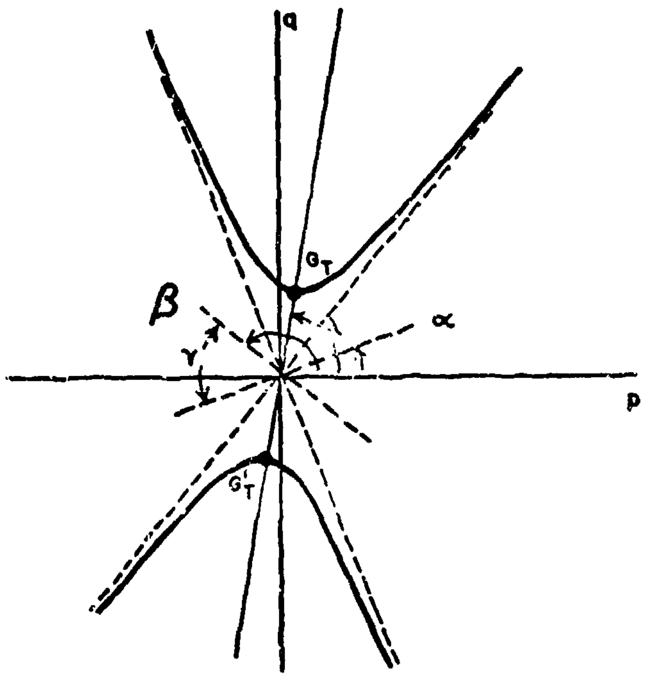


Figure 3: The hyperbola determined by a skewed symmetry

The axis of the hyperbola is the bisector of the obtuse angle made by α and β. The asymtotes make the same angle as the acute angle made by α and β.

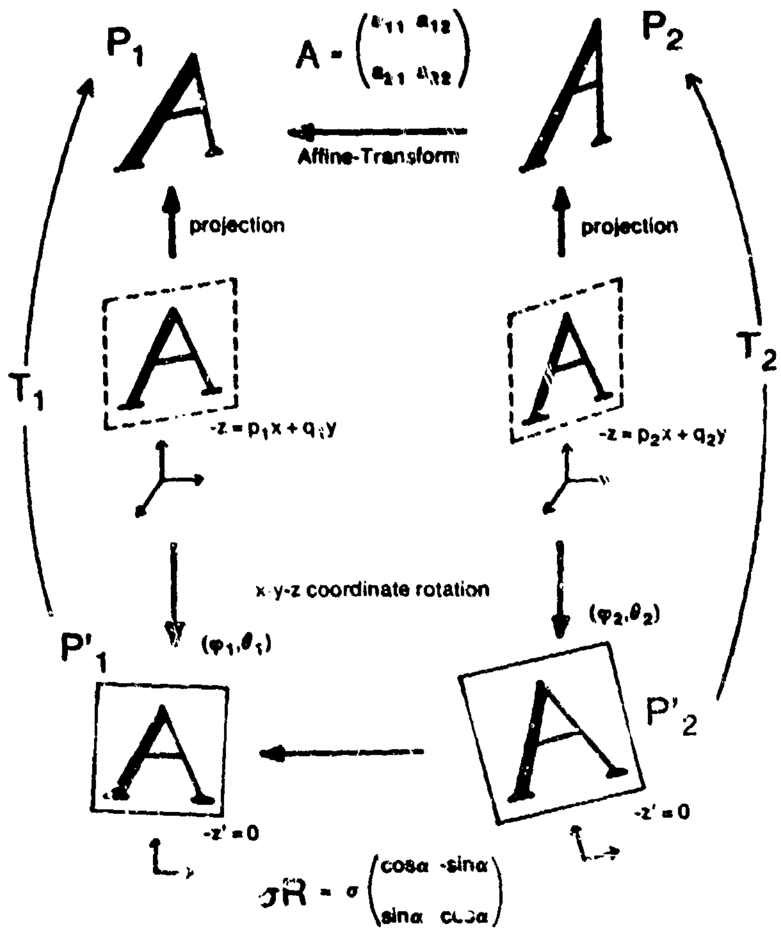
repetition a texture is defined. Suppose we have a pair of texel patterns in which one is a 2-D affine transform of the other; we call them a pair of *affine-transformable* patterns. Let us assume that *"A pair of affine-transformable patterns in the picture are the projection of similar patterns in the 3-D space (i.e., they can be overlapped by scale change, rotation, and translation)"*. The above assumption can be schematized by Figure 4.

Then we can derive a relationship between the gradients G_1 and G_2 of the scene constituents of the texel. This constraint is determined solely by the matrix A, which is determined by the relation between P_1 and P_2 which is *observable* in the picture: without knowing either the original patterns (P'_1 and P'_2) or their relationships (σ and R) in the 3-D space.

The constraints by the affine transformable patterns can be used for shape recovery of an object, like Figure 5, on whose surface a number of patterns are printed or stamped such as textile or wall papers. The assumptions we used for the skewed symmetry, the affine-transformable patterns, and texture analysis can be generalized as: *"Properties observable in the picture are not by accident, but are projections of some preferred corresponding 3-D properties."* This provides a useful meta-heuristic for exploiting image properties: we can call it the meta-heuristic of *non-accidental image properties*. Methods of aggregating many local constraints into consistent shapes are being further studied.

3-D shape sensing and analysis

At CMU, optical noncontact ranging devices for medium (50 cm) and short (5 cm) range are being developed [1]. Both use analog position sensor chips to detect the location of an intensity spot which is projected on the object surface by a light source. Unlike vidicon or CCD array sensors, scanning the field of view is not necessary. The chip outputs currents which, with a little computation, tell the location of the spot. This can provide simple, fast, accurate, noncontact visual sensing of range information. Prototype devices were constructed for both medium and proximity ranging devices. They are being tested presently and will be used mainly for robotic applications. We expect the performance of 10,000 points/sec. and 1,000 line resolution. In parallel with the development of ranging devices, we are developing programs to reliably extract local surface orientations from range data. Based on the theoretical results of Kender, we are looking at the Gaussian sphere representation for object surface orientations using the sinusoidal mapping of a sphere to a 2-D plane.



P_1
$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$
P_2
Affine-Transform
projection
projection
T_1
T_2
$-z = p_1x + q_1y$
$-z = p_2x + q_2y$
x-y-z coordinate rotation
(φ_1, θ_1)
(φ_2, θ_2)
P'_1
P'_2
$-z' = 0$
$-z' = 0$
$\sigma R = \sigma \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$

Figure 4: A schematic diagram showing the assumption on the Affine transformable patterns.

Consider two texture patterns P_1 and P_2 in the picture, and place the origins of the x-y coordinates at their centers, respectively. The transform from P_2 to P_1 can be expressed by a regular 2x2 matrix $A = (a_{ij})$. P_1 and P_2 are projections of patterns P'_1 and P'_2 which are drawn on the 3-D surfaces. We assume that P'_1 and P'_2 are small enough so that we can regard them as being drawn on small planes. Let us denote the gradients of those small planes by $G_1 = (p_1, q_1)$ and $G_2 = (p_2, q_2)$, respectively; i.e., P'_1 is drawn on a plane $-z = p_1x + q_1y$ and P'_2 on $-z = p_2x + q_2y$.

Integrated System of Database and Photo Interpretation

A system for photo interpretation tasks needs more than simple image interpretation techniques. Newly acquired images have to be assimilated into the system, compared with existing images and maps, and interpreted by using the existing relevant information as knowledge. The extracted information then updates the database. At CMU, we are currently developing a demonstration system MAPS which integrates image database, map representation, interactive image manipulation and display, and automatic image analysis [9] [8]. Figure 6 shows the present configuration of MAPS.

Multi sensor, multi data-type image database

We currently have multi-data type, multi-sensor image database for the Washington D.C. area bounded by <North 38°, West 76°> and <North 39°, West 78°>. It includes:

- Sensory Images

o airborne	monochromatic (BW)	2048x2048
o Skylab	color (COL)	2300x2300
o airborne	color infrared (CIR)	2200x2200
o Landsat	multispectral (MSS)	3200x2400

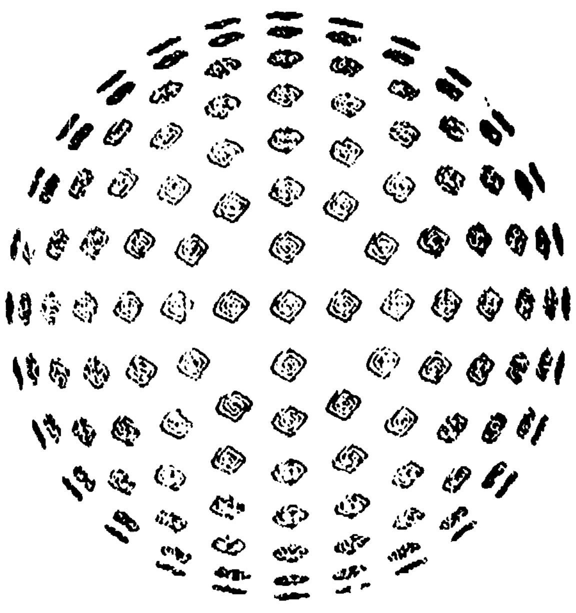


Figure 5: A picture of a ball with identical patterns stamped

- Terrain Data
  - o DLMS Digital Terrain Data File: 100 m x 100 m/pixel, 1 m resolution in altitude
- Map Data
  - o DLMS Cultural Feature Data File: approx. 18,000 features (point, linear, area)
- Symbolic Map Representation
  - o We are developing a symbolic representation of map (2D and 3D): interconnection of roads, buildings (shape, height), etc.

Manipulation, generation and display of images

Over the past year, we have implemented a large number of facilities to manipulate, to generate, and to display images using the database. The representative capabilities are: (See [8] for details)

Browse:
This is a very general, flexible multi-window image display using a Grinnell color display.

Hand Segmentation:
An image can be segmented interactively, and the results are stored as part of the image description file.

Terrain Data Manipulation and Display:
Digital terrain data can be manipulated and displayed to show 3-D representation, contours and 3-D features.

Landmark Extraction:
The landmark file contains names, descriptions and image chips of a large number of landmarks within the task area. Knowing grossly the area that a new image covers, a user can quickly locate landmarks in it in order to establish a correspondence between image and map.

Correspondence Coefficients Computation:
This program computes coefficients of the first, second and third order polynomial transforms between map and image.

Intervisibility Map:
This is being developed as one of the planned CMU contributions to the Testbed. Moravec [10] is developing a program which will be able to rapidly generate views of three-dimensional scenes described by large numbers of planar faces. The techniques used can represent the effect of shadows and produce intervisibility maps as special cases.

His method relies on a new hidden surface removal algorithm of Fuchs which generates views in linear time. (A somewhat more expensive view-independent presort must be done just once for the scene.) It generates intermediate images which contain high resolution views of the scene, but which only indicate which face is visible at each pixel, not its brightness. Because these *face identity pictures* consist of large conti[illegible]us regions with the same value they can be represented very efficiently in *quad tree* structures which recursively subdivide along edges, but represent large constant areas as single nodes. The cost of both computing and storing the face identity quad trees grows as O(*n*log*n*) with the picture resolution (and edge length), allowing very high resolutions; the cost of a conventional raster grows quadratically with the resolution.

Image analysis and interpretation

Segmentation

A region segmentation program [11] has been reimplemented in the C language on a VAX 11/780 under UNIX. This is also one of our Testbed contributions and will soon be delivered to SRI. This segmentation program has all the features of the Ohlander-Price-Shafer segmentation method. In addition, it will include enriched capabilities for manual/automatic image analysis (region analysis, region representation, histogram evaluation, threshold selection).

Image Registration and Stereo Analysis

Lucas [7] presents an iterative image registration technique, a type of Newton-Ralphson iteration, that uses spatial intensity gradient information to direct the search for the position of the best match. This technique takes advantage of the fact that in many applications the two images are already in approximate registration. It can be generalized to deal with arbitrary linear distortions of the image, including rotation.

An analysis on convergence suggests that convergence is obtained if the initial estimate is within a distance of one half of the size of the object. Range of convergence can be expanded by first smoothing the image. In fact, since frequency-limited images (low-pass or band-pass filtered) can be sampled at lower resolution, we can adopt a coarse-fine strategy together with this iterative registration.

This iterative registration algorithm with coarse-fine strategy was applied to stereo matching problem. Using the same principle, iterative formulas were derived for *both* the disparity and for the camera model. They were tested with real images.

Feature Extraction

Automatic stereo matching and interpretation of urban scenes include difficult and new problems. In stereo, for example, large discontinuities of disparity must be explicitly taken into account. For this, it is important first to extract cultural features, such as edges and corners of buildings. Stereo matching can be done using those features. Photographs of urban areas have common

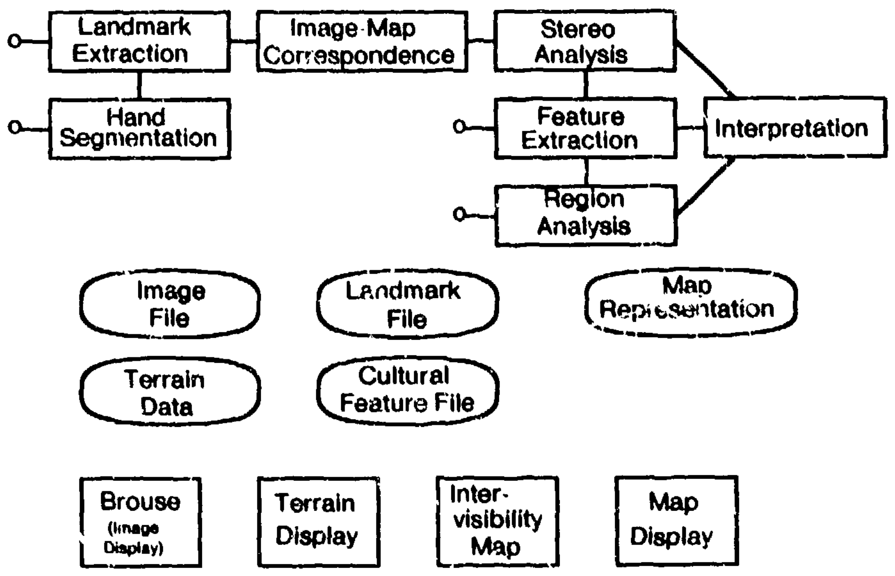


Figure 6: Present configuration of MAPS

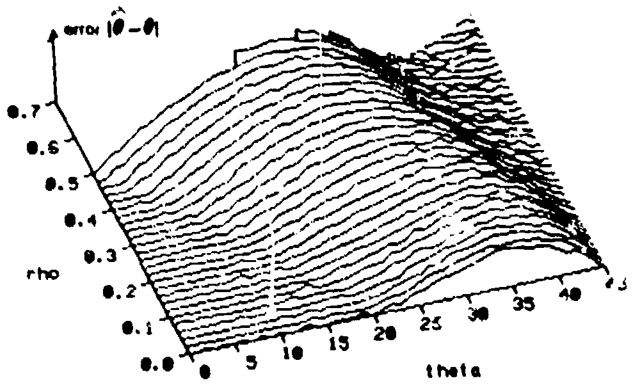
characteristics: they include a large of number of linear features; there are a few dominant orientations of the linear features due to the fact that buildings and roads are mostly aligned and rectangular; and the features tend to be small and cluttered (e.g., two parallel lines can be very close).

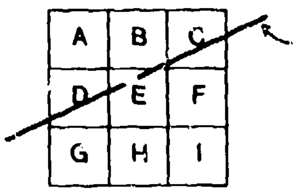
To increase the accuracy of edge operators, their window size could be enlarged, but this often results in failure to detect tiny features. Deviation of the detected orientations of edges by a 3x3 Sobel operator from true orientations was carefully analyzed. The result is shown in Figure 7. It illustrates how the operator gives better accuracy when applied to edges aligned with the picture axes. However, those directions are not known beforehand, and vary from part to part of the image.

First, the image is divided into overlapping small areas (about 100x100). The θ histogram of the Hough transform on each small area detects dominant orientations of edges in that area. For each orientation, we recompute the more precise edge orientation by locally rotating the images. Then, the ρ histogram is taken using the revised θ's to separate individual line features. This method allows the extraction of very close parallel lines, such as the case that one upper roof edge almost overlaps another lower roof edge. We are continueing this effort toward precise extraction of important micro features in urban scenes. Such an extraction procedure will form a basis for stereo matching and interpretation of urban scenes.

Conclusion

At CMU we are continuing to develop theories of shape understanding. As well as basic theoretical issues, we expect that our robotic applications will stimulate the practical aspects of the theory. We expect to continually extend our integrated database-supported photo interpretation system. Demonstration of how symbolic map representation can successfully interact with aerial photo interpretation (map-guided interpretation and map updates) is one of our main goals. For the Testbed activities, the region segmentation program and the fast generation of intervisibility map will be soon deliverable, and implementation of Moravec's stereo on VAX will begin.





$$\rho = x\cos\theta + y\sin\theta$$

$$\Delta x = (C+2F+I) - (A+2D+G)$$
$$\Delta y = (A+2B+C) - (G+2H+I)$$

$$\hat{\theta} = \mathrm{arctan2}(\Delta y, \Delta x)$$

Figure 7: Analysis of the error in the edge orientation measured by a 3x3 Sobel operator

A simple step edge is assumed to pass the central pixel. The display shows how the error $|\hat{\theta} - \theta|$ depends on ρ and θ. Naturally, it is small when the edge passes the three horizontal (or vertical) pixels.

References

[1] Kanade, T. and Asada, H.
Noncontact Visual 3-D Sensing Devices.
In *SPIE Technical Symposium East '81 (to be presented)*. Society of Photo-Optical Instrumentation Engineers, 1981.

[2] Kanade, T. and Kender, J. R.
Mapping Image Properties into Shape Constraints: Skewed Symmetry, Affine-Transformable Patterns, and the Shape-from-Texture Paradigm.
Technical Report CMU-CS-80-133, Carnegie-Mellon University, July, 1980.

[3] Kanade, T.
A Theory of Origami World.
Artificial Intelligence 1:279-311, 1980.

[4] Kanade, T.
Recovery of the Three-Dimensional Shape of an Object from a Single View.
Artificial Intelligence (to appear), 1981.

[5] Kender, J. R.
Shape from Texture.
PhD thesis, Carnegie-Mellon University, 1980.

[11] Shafer, S. A.
MOOSE Users'Manual implementation Guide Evaluation.
Technical Report IfI-HH-B-70/80, University of Hamburg, April, 1980.

[12] Stevens, K. A.
Surface Perception from Local Analysis of Texture and Contour.
PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, AI-TR 512, February, 1980.

[6] Kender, J. R.
The Gaussian Sphere: A Unifying Representation of Surface Orientation.
In *Proc. Image Understanding Workshop*, pages 157-160. 1980.

[7] Lucas, B. L. and Kanade, T.
An Iterative Image Registration Technique with an Application to Stereo Vision.
(in this volume).

[8] McKeown, D. Jr. and Kanade, T.
Database Support for Automated Photo Interpretation.
(in this volume).

[9] McKeown, D. Jr.
Knowledge Structuring in task Oriented Image Databases.
In *Workshop on Picture Data Description and Management*, pages 145-151. IEEE, August, 1980.

[10] Moravec, H.
Internal memo.

UNDERSTANDING FEATURES, OBJECTS, AND BACKGROUNDS
PROJECT STATUS REPORT 1 APRIL 1980-31 JANUARY 1981
CONTRACT DAAG-53-76C-0138 (DARPA ORDER 3206)

Azriel Rosenfeld
Principal Investigator

Computer Vision Laboratory, Computer Science Center
University of Maryland, College Park, Maryland

ABSTRACT

Current activities on the project are reviewed under the following headings:

1) Segmentation
2) Local feature detection
3) Feature linking
4) Hierarchical representation

1. INTRODUCTION

This project is concerned with the study of advanced techniques for the analysis of reconnaissance imagery. It is being conducted under Contract DAAG-53-76-C-0138 (DARPA Order 3206), monitored by the U.S. Army Night Vision and Electro-Optics Laboratory, Ft. Belvoir, VA (Dr. George Jones). The Westinghouse Systems Development Division, under a subcontract, is collaborating on implementation and application aspects.

The previous phase of the project, entitled "Image Understandi Using Overlays", was concluded during the past reporting period. Accomplishments under this phase are summarized in a Final Report dated May 1980 [1], which also contains a bibliography of all reports and papers produced during this period.

The current phase of the project is concerned with three principal areas: (a) comparative analysis of segmentation techniques applied to FLIR imagery; (b) development of an inference-based approach to target detection on FLIR imagery; and (c) optical flow analysis of time-varying imagery. Work in area (b) is in progress and will be described in forthcoming technical reports. Area (a) has emphasized methods based on hierarchical ("pyramid") image representations, some of which are reviewed in this report and in two separate papers in these Proceedings [2,3]. Other separate papers [4,5] deal with some of the work done in area (c). In addition, the project is preparing software contributions to the DARPA/DMA Image Understanding Testbed; the first of these will be a general-purpose software package for implementing relaxation processes at the pixel level.

This report reviews activities on the project during the period April 1980-January 1981. This work is covered under the headings of segmentation; local feature detection; feature linking; and hierarchical representation. The work is summarized only briefly, since it is covered in greater detail in individual technical reports and Image Understanding Workshop papers.

2. SEGMENTATION

2.1 Color pixel classification

When pixels in a black-and-white image are classified by thresholding their gray levels, gradient magnitude information can be used in various ways as an aid in threshold selection. In particular, a histogram of the gray levels of pixels whose gradient magnitudes are low has sharper peaks and deeper valleys than the histogram of the entire image, since the low-gradient pixels tend to come from the interiors of regions, not from region border zones; it is easier to choose useful thresholds (at valley bottoms) from this improved histogram. Analogously, when pixels in a color or multispectral image are classified on the basis of their spectral signatures, the color gradient magnitude can be used as an aid in defining decision surfaces that separate clusters of pixels having like signatures. In fact, a scatterplot of the signatures of pixels whose color gradient magnitudes are low has more clearly separated clusters than the scatter plot of the entire image, for the same reason as in the grayscale case. This phenomenon is illustrated in Figure 1. For further details and additional examples, see [6].

2.2 Mosaicking

When aerial photographs are combined into a photomosaic, seams are often apparent between the parts. These seams are caused by gray level differences due to the different conditions under which the parts were recorded. A relaxation method has been developed that generates a gray level correction function such that, when this function is subtracted from the mosaic, the seams are eliminated, but the details of the photographs are not affected. The algorithm does not assume any specific types of gray level differences among the parts, nor does it require the existence of overlaps between the parts, and it can be used for arbitrary numbers of parts; but it does have the drawback that if a seam coincides with an edge between two regions, that edge will be eliminated. The algorithm constructs a seam-eliminating function which, when subtracted from the mosaic, causes the gray levels at pairs of adjacent points on opposite sides of a seam to become equal, and which otherwise is as smooth as possible. An example of mosaic seam elimination using this algorithm is shown in Figure 2. Other examples, and further details, can be found in [7].

3. LOCAL FEATURE DETECTION

3.1 Higher-order edge detectors

One way to define edge detectors for digital images is to fit a polynomial surface to a neighborhood of each pixel, and take the magnitude of the gradient of that surface as an estimate of edgeness. The polynomial fitting process is usually carried out for symmetric neighborhoods, using polynomials of degree 1 or 2. Using least squares fitting by orthonormalization and Richardson extrapolation, one can calculate such edge estimates for other classes of neighborhoods, and for higher-order polynomial models [8].

As a further application of this approach, edge detectors can be defined based on least-squares surface fitting in which the surface is a step edge superimposed on a low-order polynomial function. This makes it possible to "filter" optimal step-based operator responses so as to discriminate against noise responses, by rejecting responses for which the fit is poor, without discriminating against low-contrast edges (which is unavoidable if thresholding is used for noise suppression). An example of such edge "filtering" is shown in Figure 3. For other examples, and further details, see [9].

3.2 Edge evaluation

A method of evaluating edge detector output has been developed, based on the local good form of the detected edges. It combines two desirable qualities of well-formed edges -- good continuation and thinness. The measure has the expected behavior for known input edges as a function of their blur and noise. It yields results generally similar to those obtained with measures based on discrepancy of the detected edges from their known ideal positions, but it has the advantage of not requiring ideal positions to be known. It can be used as an aid to threshold selection in edge detection (pick the threshold that maximizes the measure), as a basis for comparing the performances of different detectors, and as a measure of the effectiveness of various types of preprocessing operations facilitating edge detection.

This method is described in detail in a separate paper in these Proceedings, where examples of its performance are also given [10].

4. FEATURE LINKING

4.1 Edge segment linking

A system of programs that links edge segments based on both gray level and geometric criteria has been developed and applied to the detection of buildings and roads on aerial photographs. Preliminary results using these programs were described in [11]; a more detailed description, and numerous additional results, are presented in [12]. Further work along these lines led to the development of figures of merit for linking compatible segments (i.e., segments that could be consecutive sides of an object) and antiparallel segments (i.e., segments that could be opposite sides). For compatible pairs, the figure of merit is based on the geometrical configuration of the segments, the similarity of the gray levels on their "object" sides, and the similarity between their object sides and the line joining their endpoints. For antiparallel pairs, it is based on the homogeneity of gray level between the edges and the amount of overlap between them. These figures of merit have highly bimodal histograms, making it quite easy to decide which pairs of segments should be linked, as illustrated in Figures 4 and 5. They should be useful in the design of relaxation-like schemes for classifying edge segments. For further details, and many additional results, see [13,14].

4.2 Reconstruction from gray-weighted medial axes

A method of defining a "min-max medial axis transformation" (MMMAT) for grayscale images, based on iterated local MIN and MAX operations, was described in a previous report [1]. This transformation associates with each pixel a vector of gray level increments, and exact reconstruction of the image is possible from these vectors. Moreover, good approximations to the image can be reconstructed using only the strongest components of the strongest few vectors. A few illustrations of this were given in an earlier report; further details and additional examples can be found in [15].

5. HIERARCHICAL REPRESENTATION

Extensive work has been done on this project on the use of pyramid and quadtree structures for image representation and processing. The work done in this area through March 1980 was summarized in [16]. In this section we briefly summarize developments in this area during the past reporting period.

5.1 Quadtree-to-raster conversion

An algorithm for converting quadtree representations of binary images to row-by-row (e.g., run-length) representations was described and partially analyzed in an earlier report. More recently, a comparative study and complete analysis of four such algorithms has been conducted [17]. The simplest algorithm is a straightforward top-down approach that visits each run in a row in succession starting at the root of the tree; the other algorithms proceed in a manner akin to an inorder tree traversal. The analysis shows under what circumstances each algorithm is preferable. They have all been shown to have execution times proportional to the sum of the heights of the blocks comprising the image.

5.2 Quadtree-based image smoothing

Two methods for smoothing an image using quadtree approximations to the image have been developed. One uses the sizes of the leaves in the quadtree to determine neighborhood sizes over which to apply the smoothing. The other method maps each image gray level i into the gray level j into which i most frequently maps when we replace the level of each pixel by the level of the quadtree leaf to which it belongs. Results obtained using these methods, as well as a local histogram peak sharpening method, are shown in Figure 6. The second quadtree-based method seems to give the best results. Additional examples, and detailed descriptions of the methods, can be found in [18].

5.3 Edge pyramids and quadtrees

An edge (or curve) pyramid is a sequence of successively lower-resolution versions of an image, each containing a summary of the edge information in its predecessor. This summary includes the average edge magnitude and direction in each "block" of the higher-resolution image, together with an intercept in that block and a measure of the error in the direction estimate. An edge quadtree, analogously, is a variable-resolution representation of the edge or curve information in the given image, constructed by recursively splitting the image into quadrants based on magnitude, direction, intercept, and error information. Advantages of these representations include their registration with the original image, their ability to represent many edges or curves in a single tree structure, and their ability to perform many operations on the represented data efficiency. A detailed description of these representations, together with examples, can be found in a separate paper in these Proceedings [2].

5.4 Pyramid linking

When an image is smoothed using small blocks or neighborhoods, the results may be somewhat unreliable due to the effects of noise on small samples. When larger blocks are used, the samples become more reliable, but they are more likely to be mixed, since a large block will often not be contained in a single region of the image. A compromise approach is to use several block sizes, representing versions of the image at several resolutions, and to carry out the smoothing by means of a cooperative process based on links between blocks of adjacent sizes. These links define "block trees" which segment the image into regions, not necessarily connected, over which smoothing takes place. The basic "pyramid linking" scheme was described in an earlier report. Further experiments with this scheme have led to some improvements over the original method, based on better ways of initializing the process and measuring the link merit. A detailed description of these experiments and their results can be found in a separate paper in these Proceedings [3]. It has also been found that forced-choice linking of blocks to larger blocks is not necessary; one can use weighted links, recomputing the weights at each iteration, and it turns out that the weights converge to 0's and 1's as the process stabilizes. Generalizations of this approach to image features other than gray level, including color signatures and textural properties, have also been investigated and will be described in future reports.

References

1. A. Rosenfeld, Image Understanding Using Overlays, Final Report, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, May 1980.

2. M. Shneier, Two Hierarchical Linear Feature Representations: Edge Pyramids and Edge Quadtrees, TR-961, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, October 1980; also appears in these Proceedings.

3. T. Silberberg, S. Peleg, and A. Rosenfeld, Multi-Resolution Pixel Linking for Image Smoothing and Segmentation, TR-977, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, November 1980; alro appears in these Proceedings.

4. K. Prazdny, Relative Depth and Local Surface Orientation from Image Motions, TR-996, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, January 1981; also appears in these Proceedings.

5. K. Prazdny, Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinearly Moving Observer, TR-1009, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, February 1981; also appears in these Proceedings.

6. A. Broder and A. Rosenfeld, Gradient Magnitude as an Aid in Color Pixel Classification, TR-906, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, June 1980.

7. S. Peleg, Elimination of Seams from Photomosaics, TR-895, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, April 1980.

8. D. G. Morgenthaler, Feature Detectors Based on Higher Order Polynomials, TR-896, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, April 1980.

9. D. G. Morgenthaler, A New Hybrid Edge Detector, TR-897, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, April 1980.

10. L. Kitchen and A. Rosenfeld, Edge Evaluation Using Local Edge Coherence, TR-981, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, December 1980; also appears in these Proceedings.

11. M. Tavakoli, Toward the Recognition of Cultural Features, Proc. DARPA Image Understanding Workship, April 1980, 33-37.

12. M. Tavakoli and A. Rosenfeld, Toward the Recognition of Buildings and Roads on Aerial Photographs, TR-913, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, July 1980.

13. M. Tavakoli and A. Rosenfeld, A Method for Linking Pairs of Compatible Linear Features, TR-930, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, August 1980.

14. M. Tavakoli and A. Rosenfeld, A Method for Finding Pairs of Anti-Parallel Linear Features, TR-943, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1980.

15. S. Wang, A. Y. Wu, and A. Rosenfeld, Image Approximation from Grayscale "Medial Axes," TR-900, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, May 1980.

16. A. Rosenfeld, Some Uses of Pyramids in Image Processing and Segmentation, Proc. DARPA Image Understanding Workshop, April 1980, 112-120.

17. H. Samet, Algorithms for the Conversion of Quadtrees into Rasters, TR-979, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, November 1980.

18. S. Ranade and M. Shneier, Using Quadtrees to Smooth Images, TR-894, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, April 1980.

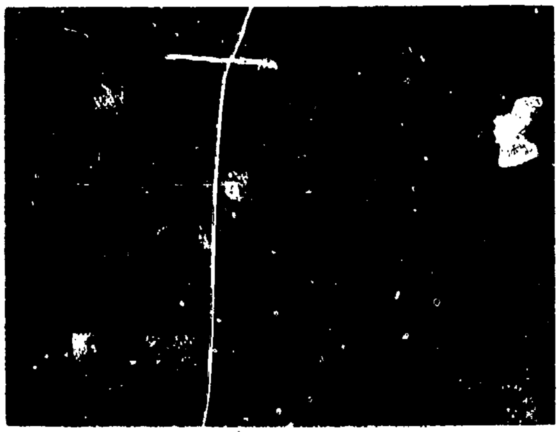
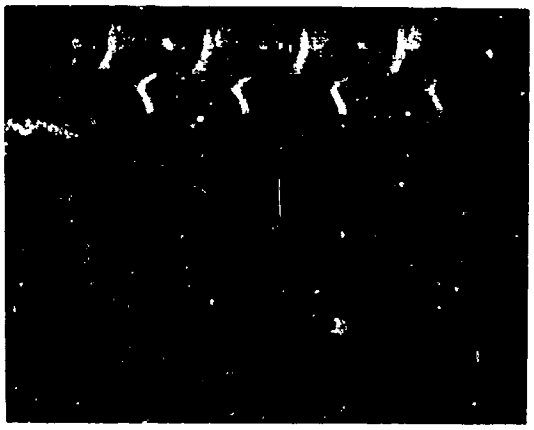


Figure 1. Scatterplot enhancement by suppression of high-gradient pixels.

key to parts:

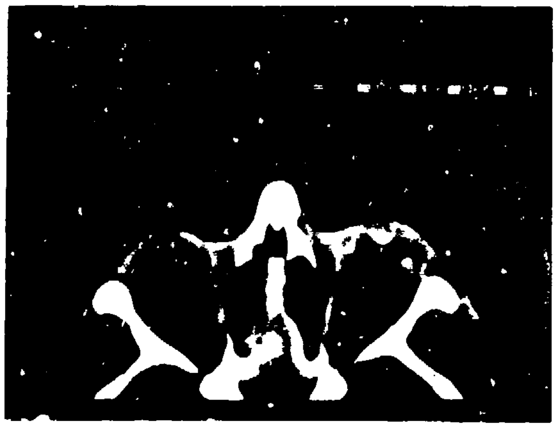
g	h	i
d	e	f
a	b	c

a-b) Two bands
c) Scatterplot of (a) vs. (b), log scaled
d-e) Edge responses in the two bands (RMS Roberts operator)
f) Color edge response: RMS of (d) and (e)
g) Enhanced scatterplot, log scaled; pixels with edge responses > 2 have been suppressed
h) Mask showing suppressed pixels
i) Histogram of edge responses, log scaled

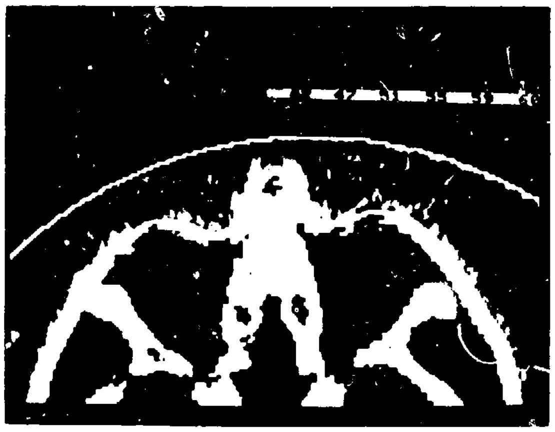


(a) (b) (c) (d)

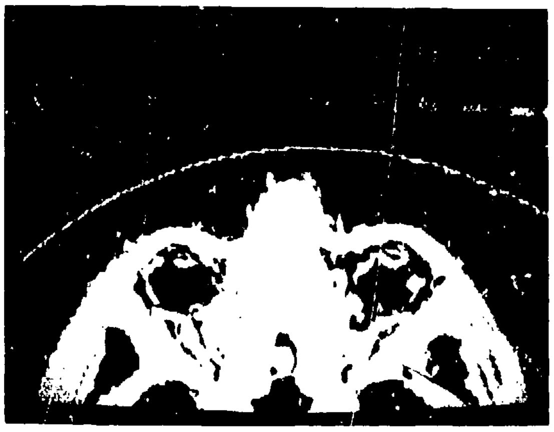
Figure 2. Mosaic seam elimination by relaxation: (a) Original (b-d) After 50, 100, and 200 iterations



(a)



(b)



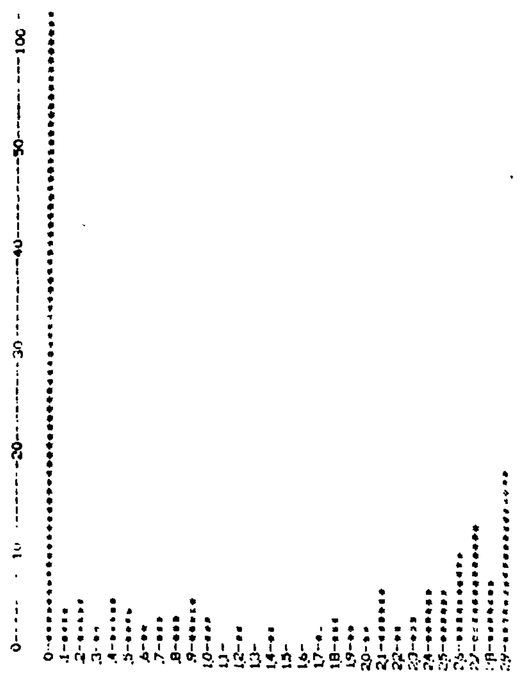
(c)

Figure 3. Suppression of low-fit edges

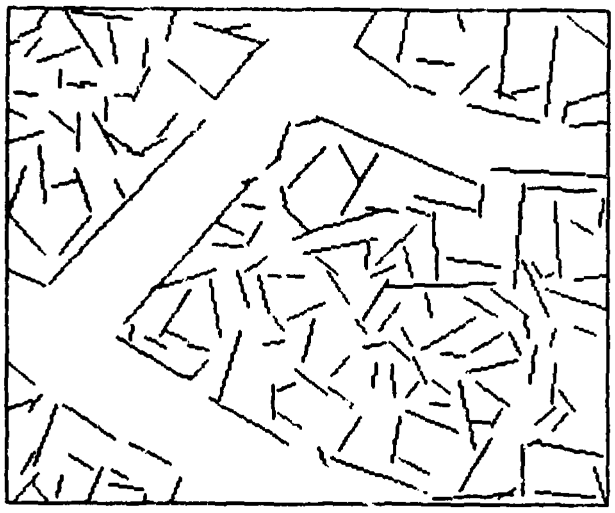
a) Original (skull cross-section)

b-c) High-fit edges (two-thresholds). Note that the lower threshold brings out the low-contrast edges without affecting the noise edges outside the skull.

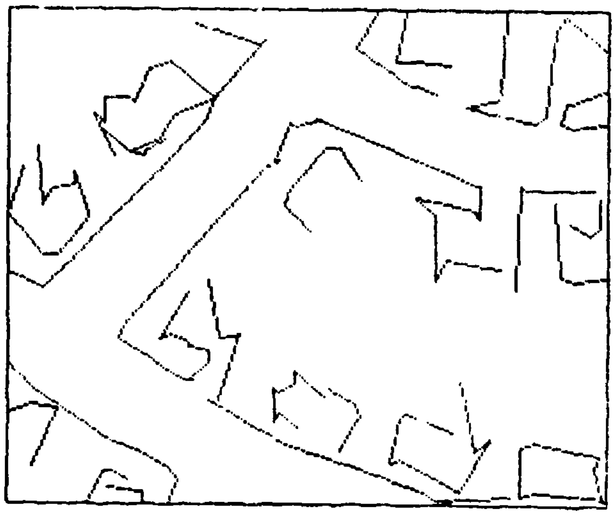
(a) Surburban scene



(c) Compatibility merits for pairs of segments in (b)

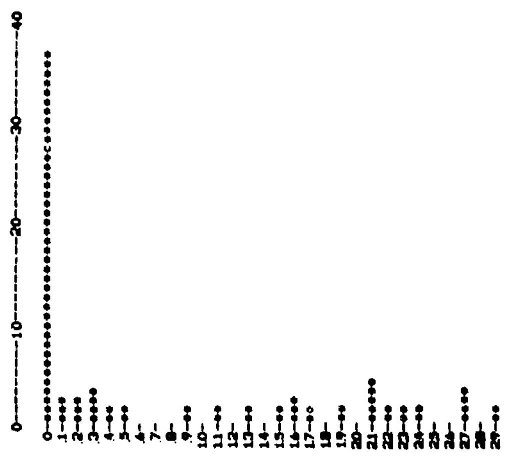


(b) Edge segments extracted from (a)

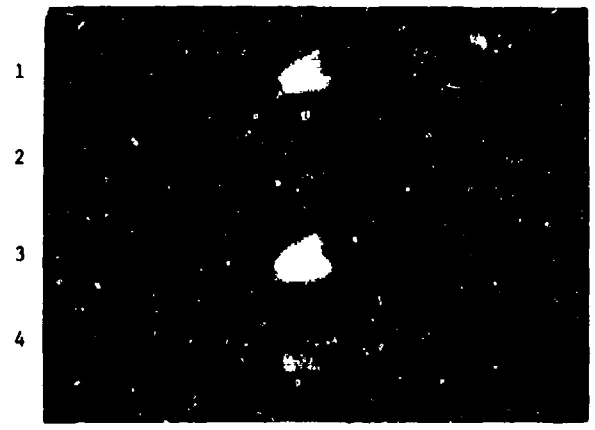


(d) Pairs of segments having compatibilities ≥ 2

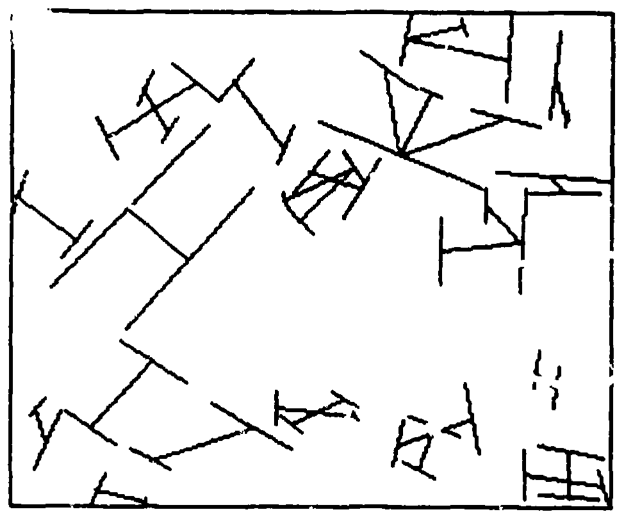
Figure 4. Compatibility merit for edge segments



(a) Antiparallelness merits for the segments in Figure 4b



a 1) Input image
2) Results of histogram sharpening
3) Results of variable-neighborhood smoothing
4) Results of smoothing using most frequent leaf value



(b) Pairs of segments having antiparallelness merit ≥ 2.

(b) Histograms of the images in (a)

Figure 5. Antiparallelness merit for edge segments

Figure 6. Quadtree-based image smoothing

MIT PROGRESS IN UNDERSTANDING IMAGES

Michael Brady

The Artificial Intelligence Laboratory
Massachusetts Institute of Technology

In this series of Image Understanding Workshop Proceedings, we have stressed the issue of representation. In particular, we have described the development by Horn and his collaborators of the reflectance map *and the* albedo image, *and we have described the work of Marr and his collaborators using the* primal sketch, *the* 2 1/2-D sketch, *and axis-based 3-D models as part of a comprehensive theory of recognition.*

In the April, 1980 Proceedings, we reviewed work on texture gradients, zero crossings, and atmospheric modelling. We introduced Horn and Schunck's work on the determination of optical flow fields from smoothly varying brightness patterns.

Here we review work on using occluding boundaries *to facilitate the computation of shape from shading, the* interpolation of smooth surfaces *from a discrete set of points, the* detection and perception of motion, vision hardware, *and the* geometric relations *made explicit in the full Primal Sketch.*

Introduction

Practical applications of Image Understanding require the development of algorithms to perform processes such as extracting the important intensity changes from a scene, or detecting movement in an image and interpreting it as motion in space. One approach is to develop algorithms that are tailored from the start to a particular application domain. An alternative is to understand the basic principles underlying each such module. One may then be in a position to apply substantially the same ideas in situations as diverse as remote sensing, object recognition, and object tracking. We have followed the latter course. One important process is the determination of surfaces from images. This is the goal of stereo, shape from shading, shape from texture, and shape from motion. Progress on this problem, including the interpolation of smooth surfaces from a discrete set of *boundary points* is a recurrent theme in the current report.

Shape from shading and occluding boundaries

Horn and his collaborators have devoted considerable attention to computing the shape of a visible surface from the intensities that comprise its image. The relationship between them is expressed mathematically by the *Image Irradiance Equation*, which is a first order partial differential equation of the form

$$I(x,y)=R(p,q),$$

where p and q are is suitable pair of parameters that specify the local surface normal. One such pair is the gradient of the depth function z from the observer with respect to the image coordinates x and y. The function R encodes the reflectance characteristics of the surface and the distribution of light sources, both of which are assumed to be unknown but fixed (see Horn 1975, Horn and Sjoberg 1980). Horn(1977) introduced the *Reflectance Map* which associates with each surface orientation (p,q) of Gradient Space a scaled value for the image intensity R(p,q). The Reflectance Map has proved to be a valuable representation in several diverse applications of shape from shading (Woodham 1978, Sjoberg and Horn 1980, Silver 1980, Ikeuchi 1981).

The earliest algorithm for computing shape from shading was devised by Horn(1975). It exploited the *characteristic strip* method of reformulating a single partial differential equation as a set of five ordinary differential equations. The idea is to determine the shape of a surface by computing a set of space curves, called characteristics, which are everywhere tangential to the surface. Horn(1975, 1977) derived an iterative algorithm to find characteristic strips. It starts from a point (x_0,y_0) in the image at which the surface gradient (p_0,q_0) is known. The step from (x_n,y_n) to $(x_{(n+1)},y_{(n+1)})$ is in the direction of the normal to the iso-brightness contour passing through (p_n,q_n) in the Reflectance Map. Similarly, the incremental change in gradient from (p_n,q_n) to $(p_{(n+1)},q_{(n+1)})$ along the characteristic is in the direction of the normal to the iso-brightness contour passing through (x_n,y_n) in the image.

One problem with this method concerns the choice of the *singular image point* (x_0,y_0) required to start the iterative process at which the surface gradient (p_0,q_0) is determined uniquely by the intensity data. A further problem is that Horn's algorithm depends on the assumption that the underlying surface is locally convex at the singular point. Finally, the class of Image Irradiance Equations for which Horn's algorithm works was unknown. (The latter question has recently been answered by Bruss(1981).) Consequently research was directed to discover the criteria under which the shape of a surface is uniquely determined by an image. One suggestion was that bounding or occluding contours provide such conditions. Along such contours the surface normal can be computed exactly from the image. However, occluding contours pose a problem for the gradient parameterisation of local surface orientation, namely at least one of the gradients p or q is infinite. Ikeuchi and Horn(1981) propose a different parameterisation of surface orientations that corresponds to *Stereographic Projection*. Whereas Gradient Space is the planar projection of the Gaussian Sphere from its center, the Stereographic Projection is from the north pole (it is assumed that the viewpoint is from the south pole facing the center of the Gaussian Sphere).

Ikeuchi and Horn(1981) note some additional problems with the characteristic strip method for solving the Image Irradiance Equation. First, since the method proceeds unidirectionally along the strip, the method cannot exploit boundary conditions at *both* ends of the strip. Second, the build up of numerical errors along any individual strip can be substantial. The alternative method devised by Ikeuchi and Horn is to formulate a smoothness condition in terms of the Stereographic parameterisation and use it as the basis of a local parallel computation which "fills in" local surface normals from the known values on the boundary to the unknown interior. The resulting algorithm has been tested on a variety of images and works well. In particular, it appears to degrade gracefully as errors are introduced to the placement of the light source, the surface orientation on the boundary, and the nature of the reflectivity assumed for the surface. Strong *empirical* evidence is provided that the algorithm converges, although no proof is demonstrated. In case the occluding contour is partially incomplete, Ikeuchi and Horn's algorithm still appears to converge, though it is not known at how many points it is necessary to specify the (Stereographic parameterisation of the) surface normal.

Bruss(1980) has recently studied some of the mathematical properties of the Image Irradiance Equation. First, she has shown that discontinuous solution surfaces can arise from a continuous Image Irradiance Equation. It follows that one cannot determine for a continuous Image Irradiance Equation whether or not there is an edge. The curvature of a surface also *cannot* be determined from its image. As an example, the Image Irradiance Equation $x^2+y^2=p^2+q^2$ has two different solution surfaces, one of which consists entirely of hyperbolic points, while the other consists entirely of elliptic points. However, Bruss has proved that there is only one solution that is convex. She has also shown that bounding contours can be determined from the image only when the Image Irradiance Equation is *singular*. This means that the reflectance function R and its first order partial derivatives are continuous, while the intensity function I is singular in x and/or y. For any given singular Image Irradiance Equation the points on the occluding contour can be found by inspection of the intensity function I(x,y).

Bruss has studied singular "eikonal" Image Irradiance Equations that are of the form $p^2+q^2=I(x,y)$. If the intensity function I(x,y) *vanishes to second order* at the singular point, that is to say has the form $\alpha x^2+\beta xy+\gamma y^2+O(|x|^3+|y|^3)$, then there is exactly one positive locally convex solution surface in the neighborhood of the singular point. This result is applied to show that if there is a closed bounding contour, the solution surface is unique (up to translation along the z axis). If either the reflectance function is not p^2+q^2, the intensity function does not vanish precisely to second order, or there is not a smooth closed bounding contour, there is not a unique solution surface.

Progress on shape from shading means that we can compute the topography of the visible surfaces from an image by a local parallel computation which is naturally implemented in hardware. It exploits reliable information, and, as a result of the theoretical developments sketched above, we can reasonably predict the behavior of the algorithm in unpredictable situations.

The detection and perception of motion

In the last workshop Proceedings we sketched an algorithm to determine *optical flow*. Optical flow is the distribution of velocities of apparent movement caused by smoothly changing brightness patterns. It has been noted that optical flows encode rich information about a scene and observer motion, and it has been suggested that this information can be computed from the flow field. In particular, it has been proposed that optical flow facilitates object segmentation, computation of the parameters of the observer's own motion relative to the scene, and the determination of visible local surface normals.

For the most part, research has been concerned with *interpretation*. It is generally supposed that the flow is given, that it is somehow computed automatically and sufficiently noise-free by "velocity sensitive neurons" in animate visual systems. Horn and Schunck(1981) have studied the generation of the optical flow from smoothly time varying brightness patterns. They restrict attention to imaging a flat surface, uniform incident illumination, and smoothly varying reflectance. With these assumptions, the image brightness at point (x,y) at time t is governed by

$$E_x u + E_y v = -E_t,$$

where (u,v) is the optical flow. This linear constraint specifies the component of the flow normal to the brightness gradient. In order to compute the component of the flow along iso-brightness contours a further assumption is required. Horn and Schunck propose a measure of the *smoothness of flow*. The departure from smoothness and the error due to changing brightness are combined and used to define an iterative algorithm for computing the flow from a sequence of images.

The algorithm works well on synthetic images, especially when there are no depth boundaries. It also seems to give good results when there are depth boundaries, though the errors in the flow become significant on the boundary. Schunck is continuing to develop the algorithm to make it more generally useful. It is already clear that it is difficult to achieve the ideal noise-free flow fields assumed as input by published interpretation schemes. Consequently, we shall reconsider the interpretation of flow fields generated as the output of Horn and Schunck's algorithm.

Generally, progress in the determination and perception of motion rests on the isolation of useful representations, since motion refers to *changes* to those representations. For example, Ullman's(1978) work on the correspondence computation followed Marr's(1976) discussion of the Primal Sketch. Similarly, the determination of optical flow sketched above rests upon Horn's work on shape from shading. Progress has been made on understanding motion, based on our work with several other representations. The following paragraphs summarise our efforts.

Previous workshop Proceedings have reported our work on zero crossings. Marr and Ullman(1980) suggested that the time rate of change of $S(x,y,t)=D^2G(x,y)*I(x,y,t)$ can enable one to *detect the direction of motion* of zero-crossings. The practical importance of this approach is that in attempting to track the motion of objects, it seems reasonable to find the important intensity changes and find out what they are doing. Let T(x,y,t) denote the partial derivative of S(x,y,t) with respect to time. Marr and Ullman consider the response of S(x,y,t) and T(x,y,t) in the vicinity of an moving isolated intensity edge, thin bar, and wide bar. They show for example for an edge moving to the right, T(x,y,t) is positive at the zero crossing, while for motion to the left it is negative. Marr and Ullman propose that motion to the right can be detected by the simultaneous activity of S^+, T^+, and S^-. Here S^+ refers to the positive component of S.

Work on directional selectivity has suggested a possible VLSI implementation of the $D^2G(x,y)$ operator based on an analysis of the animate retina. Marr and Ullman find close agreement at moderate speeds between their theoretical predictions of the response of ganglion X and Y cells to simple moving stimuli and actual cell recordings from the Physiology literature (see Richter and Ullman 1980, figures 13 and 15.) Richter and Ullman have recently accounted for the discrepancy at high speeds, and generally refined the model of directional selectivity by noting that the Gaussians, whose difference approximates $D^2G(x,y)$, act like RC filters with different time constants. This causes a slight delay in the onset of the negative outer part relative to the positive central part. Richter and Ullman's predictions show remarkable agreement with cell recordings for a wide variety of stimuli.

Motion at the level of the Primal Sketch and up requires careful attention to coordinate frames. Brady and Prazdny(1980) showed how apparent or induced motion can be explained on the basis of eye tracking and local coordinate frames. Local coordinate frames were a major concern of Marr and Nishihara(1977) in their contributions to object representations based on generalized cones.

Interpolation of curves and surfaces

Many of the visual processes discussed here and in previous Proceedings compute the shape of a visible surface by finding the local surface orientation everywhere within its boundaries. This includes the work of Horn and his colleagues on shape from shading, and the computation of shape from texture investigated by Stevens(1980) and Witkin(1980). On the other hand, binocular stereo computes disparity at the *discrete set* of zero crossings. A change of coordinates can convert angular disparities to depths, but to compute the local surface normal everywhere on a visible surface it is first necessary to *interpolate a smooth surface* from the discrete set of given points. Binocular stereo is not the only module which generates an incomplete surface orientation map. Stevens(1981a) considers the interpretation of surface contours, and finds that they strongly constrain the perception of the underlying surface. Horn(1982) and Marr(1978) suggest that in addition to local surface orientation, it is advantageous to make explicit discontinuités in surface orientation and depth. It is not yet clear how surface normals should be parameterised, nor how accurately their values should be represented. Moreover, substantial advantages are likely to accrue from attaching texture and color descriptors to visible surfaces, but the details are as yet unclear.

We have studied the interpolation of a smooth surface from a discrete set of points. One possibility is to use Coons patches, and Bezier and Ferguson surfaces developed for work in Computer Aided Design (CAD) and Computer Aided Manufacture (CAM)(see Faux and Pratt 1979.) A practical difficulty with this approach stems from the spatial irregularity of the discrete set of boundary values (for example disparity values at matched zero crossings.) Consequently we have investigated surface interpolation using what we know about *human vision*, by isolating constraints which have not figured largely in the development of CAD/CAM. Essentially, two such constraints have been uncovered, and are currently receiving attention.

The first was introduced by Grimson(1981). Suppose that a smooth surface S is interpolated from a given set of boundary values. Grimson observes that the *absence* of a boundary value at (x,y) means that the gradient of S cannot change too rapidly there. Grimson has coined a suggestive slogan for this analysis: *no news is good news*. To make this observation precise, Grimson notes that Horn's work on image formation enables conditions to be derived on the zero crossings that would arise from the surface S.

The second constraint is based on the idea that, in the absence of contrary evidence, the human visual system constructs the *most conservative* curve or surface consistent with the data. We are able to interpolate smooth curves and surfaces without involving rich semantics. It also seems that the shape of the boundary plays the most significant role in determining the interpolated surface (see for example figure 2-3 in Barrow and Tenenbaum 1981.) Taken together, these ideas suggest that the interpolation process can be modelled in terms of modern control theory (see for example, Schultz and Melsa 1967). The idea is to isolate an appropriate "performance index" P and *define* the interpolated surface to be the one that minimises the integral of P subject to the boundary constraints. Clearly, one requirement on the performance index is that a minimal surface should be guaranteed to exist for any given set of boundary values. To this end, Grimson has proved the following theorem: Suppose that there exists a complete seminorm d on a space H of functions, and that d satisfies the parallelogram law. Then every non-empty closed convex subset of H contains a unique function of minimum seminorm.

A number of plausible performance indices can be evaluated in the light of this theorem. Grimson notes that the mean and Gaussian curvature do *not* satisfy the required conditions. He suggests using instead the quadratic variation, which is defined to be $f_{xx}^2+2f_{xy}^2+fyy^2$, and derives a 5 by 5 interpolation operator to compute the minimal surface. The square Laplacian also satisfies the conditions of the theorem, but is rejected by Grimson on the grounds that its null space is larger (see Grimson 1981, section 3.1.4.) Recently, Brady and Horn(forthcoming) have noted that any quadratic form in f_{xx}, f_{xy}, and f_{yy} satisfies the theorem. They have shown that the quadratic forms which are rotationally invariant form a vector space which has the square Laplacian and the quadratic variation as a basis. Since the quadratic variation has the smaller null space, it is to be prefered.

Brady, Grimson, and Langridge(1980) use an approximation to the one dimensional quadratic variation f_{xx}^2 to argue that subjective contours are cubics. The exact minimal integral curvature curve has recently been found by Horn(1981). Brady and Grimson (forthcoming) use these ideas about surface interpolation to propose that subjective contours arise from surface perception.

Real time convolution

At the last workshop we took delivery of a number of the Hughes Research Laboratories VLSI multi-function convolution chips prepared for the DARPA Image Understanding Program (see Nudd et. al. 1980.) We began to construct the hardware necessary to test and evaluate them. Specifically, a "serpentine memory" was required to buffer video output from a TV camera so that 26 successive image lines could be fed to the convolver in parallel at video rates. The VLSI circuit is designed to convolve the image with a built-in 26 by 26 difference of Gaussian mask, producing a video signal that lags the TV input by 26 lines. In the process of doing this, we found that we could design a convolver using digital TTL technology that could process one million pixels per second. Although this is slower than the speed claimed for the Hughes chip, the TTL approach has a number of advantages for us. The extensive analog circuitry necessary to operate the Hughes chip can be avoided. The all digital design provides more precision than is possible with the analog storage of weights and intensities, and digital logic is easier to debug and modify. Finally, variable sized Gaussian filters could be provided.

The design of the TTL convolver embodied two principal guidelines. First, the convolver was required to serve as the first stage of a real time implementation of the Marr-Poggio-Grimson theory of stereo vision (Marr and Poggio 1979, Grimson 1980, 1981). Eventually we hope to be able to carry out the matching process required by the theory without using large "frame buffer" memories for storing the intermediate results of matching. The freedom from having to use a frame buffer allows us to handle images with an arbitrary number of lines. The present system handles 1024 pixels per line and places no restriction on the number of lines in the image. Second, the entire system was designed as a set of modules that can be separately developed. Each module takes an array from the Lisp machine memory as input, and writes its results into another array. A consequence of this design decision is that we have been able to write software simulations of incomplete parts of the overall system on the Lisp machine and test the system as a whole throughout the development process.

The current system takes its input from a one-dimensional solid state CCD array camera in conjunction with a mirror deflection system. The current camera is a 1024-element linear array scanner. A shortcoming of such devices is their limited light sensitivity resulting from the brief integration time as the mirror sweeps over the image. The present system works well with studio lighting.

The convolution module was the central focus of the first half of our development effort because of the large computational demands it makes. For digital Gaussian convolution with a 32 by 32 mask size, a minimum of 32 multiplies are required for each pixel To maintain adequate precision, the first one-dimensional convolution requires 16 8x8 multiplies while the second requires 16 8x16 multiplies. Using TRW multiplier chips we are able to process just under one million pixels per second. Higher pixel rates should be obtainable with more parallel or analog designs such as the Hughes convolver chip.

A central issue was whether to compute the Laplacian of a single Gaussian convolution, or approximate it as the difference of two Gaussians (Marr and Hildreth 1980.) Generally, for Gaussian convolutions with a given precision, the difference of Gaussian approach offers a better signal to noise ratio because of the second-order differences that arise from computing the Laplacian. Designing two copies of the same convolution circuit is also easier, so the difference of Gaussian approach was selected.

The resulting hardware takes 8 bit pixels as input and produces signed 16 bit numbers as output. The 32x32 Gaussian mask size allows difference of Gaussian convolution masks with central positive diameters of between 2 and 12 pixels. Using array to array mode on a Lisp machine, a 1000x1000 image of 8 bit pixels can be convolved and the result stored as a 1000x1000 array of 16 bit numbers in about 20 seconds. Of this time, only 1.5 seconds is needed to convolve the image, the remainder being used for paging between disk storage and memory.

The Full Primal Sketch

There were two distinct aspects to Marr's(1976) original discussion of the Primal Sketch representation. First, the important intensity changes in the image are made explicit and described as blobs, terminations, and various kinds of scene event such as shading edge. This description was called the *Raw Primal Sketch*. Second, the *Full Primal Sketch* is a hierarchical representation that results from making explicit the local geometrical structure of the "place tokens" comprising the Raw Primal Sketch and lower levels of the hierarchy. Marr(1976) suggested a number of processes which can plausibly discover the local organisation perceived by humans. These include collinearity, parallelism, the formation of "clusters", and "theta aggregation".

For example, suppose that a striped surface texture such as ruled writing paper is occluded by some nearer surface such as a book. In the Raw Primal Sketch, the line terminations corresponding to the rulings of the paper intersecting the bounding contour of the book would be aligned. That geometric arrangement has to be discovered as a curvilinear aggregate, and a description of it written down in the Full Primal Sketch. In this way, the Full Primal Sketch aims to make explicit the geometric properties that facilitate the interpretation of the Raw Primal Sketch in terms of physical surfaces. Riley's(1981) forthcoming thesis examines the theoretical basis for the Full Primal Sketch, and shows that discontinuities in certain attributes of the Raw Primal Sketch description are reliably correlated with edges of physical surfaces while others may either be due to surface creases or physical edges. Riley's work sheds some light on the confusing psychophysical phenomena of texture discrimination.

Stevens(1981b) has worked in a complementary area, studying how collinearity among points is detected. Marr(1976) conjectured that grouping (such as curvilinear aggregation) should be performed on symbolic place tokens which mark distinguished points in the image. Stevens has developed theoretical and psychophysical evidence in support of this conjecture for human vision. The core of Marr's place token hypothesis is that grouping and aggregation processes operate not on the image, but on tokens extracted from the Raw Primal Sketch. There have been proposals that edge detection mechanisms might trigger weakly on linear groupings, as blurring a line of dots into a fuzzy line might suggest. However for edge detection to extract the collinearity that way, there should be a reliable correlation between the *physical* processes that cause collinearity and the physical processes that cause relatively low spatial frequency intensity changes in the image. That correlation is weak and unreliable, as can be demonstrated by means of the Marr-Hildreth model of edge detection (Marr & Hildreth 1980).

Altogether, our increased understanding of the full Primal Sketch should enable us to extract more reliable texture descriptors from images.

References

For an extended discussion of MIT work on Image Understanding, see Volume 2 of *Artificial Intelligence: an MIT Perspective*, edited by Patrick H. Winston and Richard H. Brown, MIT Press, Cambridge, Massachusetts, 1979.

Barrow H. G. and Tenenbaum J. M. (1981), "Interpreting line drawings as three dimensional surfaces", *Artificial Intelligence: Special issue on Computer Vision* ed. Michael Brady, Vol 16.

Brady J. Michael and Prazdny K. F. (1980), "Extra-retinal signals influence induced motion: a new kinetic illusion", AIM-580, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Brady J. Michael, Grimson W. E. L. and Langridge D. (1980), "The shape of subjective contours", *Proc. AAAI*, Vol 1, 15-17.

Bruss Anna R. (1980), "The image irradiance equation: its solution and application", PhD thesis, December 1980. To appear as TR-623, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1981.

Faux I. D. and Pratt M. J. (1979), "Computational Geometry for design and manufacture", pub. Ellis Horwood, Chichester UK.

Grimson W. E. L. (1980), "A computer implementation of a theory of human stereo vision", *Phil. Trans. Roy. Soc. Lond.*.

Grimson W. E. L. (1981), "A computational theory of visual surface interpolation", AIM-613, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Grimson W. E. L. (1981), "From images to surfaces", MIT Press, Cambridge Mass.

Horn B. K. P. (1975), "Obtaining Shape from Shading Information", in *The Psychology of Computer Vision* ed. Winston P. H., McGraw-Hill

Horn, B. K. P. (1977), "Understanding Image Intensities", in *Artificial Intelligence*, Vol 8, pp. 201-231, 1977.

Horn B. K. P. (1982), "Sequins and Quills - Representations for Surface Topography", in *Representation of 3-Dimensional Objects* ed. Bajcsy R., Springer Verlag.

Horn B. K. P. and Schunck B. G (1981), "Determining optical flow", in *Artificial Intelligence: Special issue on Computer Vision* ed. Michael Brady, Vol 16.

Horn B. K. P. and Robert W. Sjoberg, (1979) "Calculating the Reflectance Map," in *Appl. Optics*, Vol 18, 1770-1779,

Ikeuchi K. (1981), "Determination of surface orientations of specular surfaces by using the photometric stereo method", to appear in *Proc. IEEE.*

Ikeuchi K. and Horn B. K. P. (1981), "Numerical shape from shading and occluding boundaries", in *Artificial Intelligence: Special issue on Computer Vision* ed. Michael Brady, Vol 16.

Marr D. (1976), "Early processing of visual information", in *Phil. Trans. R. Soc. Lond. B*, Vol 275, 483-524.

David Marr, "Representing Visual Information," AIM-415, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1977.

Marr D. (1981) "Vision", pub. Freeman, San Francisco.

Marr D. and Hildreth E. (1980) "Theory of edge detection" in *Proc. R. Soc. Lond. B*, Vol 207, 187-217.

David Marr and Keith Nishihara, "Representation and Recognition of the Spatial Organization of Three-dimensional Shapes," *Phil. Trans. Roy. Soc. B. 275.*

Marr D. and Poggio T. (1979), "A theory of human stereo vision", in *Proc. R. Soc. Lond. B*, Vol 204, 301-328.

Marr D. and Ullman S. (1981), "Directional selectivity and its use in early visual processing", in *Proc. R. Soc. Lond. B.*

Nudd G. R., Fouse S. D., Nussmeier T. A., and Nygaard P. A. (1979), "Development of Custom-Designed Integrated Circuits for Image Understanding", in *Proceedings of the Image Understanding Workshop* ed. Lee Baumann.

Richter J. and Ullman S. (1980), "A model for the spatio-temporal organization of X and Y-type Ganglion cells in the primate retina", MIT-573 The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980.

Riley M. (1981), "Representing Image Structure", forthcoming SM thesis, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1981.

Silver W. M. (1980), "Determining shape and reflectance using multiple images", SM thesis, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980.

Sjoberg Robert W. and Horn B. K. P. (1980), "Atmospheric modelling for the generation of albedo images", in *Proceedings of the Image Understanding Workshop* ed. Baumann Lee, Science Applications.

Stevens K. A. (1980), "Surface Perception by local analysis of Texture and Contour", MIT Technical Report 512, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1980.

Stevens K. A. (1981), "The visual interpretation of surface contours", in *Artificial Intelligence: Special issue on Computer Vision* ed. Michael Brady, Vol 16.

Stevens K. A. (1981), "The detection of collinearity", AIM-621, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Ullman S. (1978), "The interpretation of visual motion", publ. MIT Press, Cambridge, Mass.

Woodham, R. J. (1978), "Reflectance Map Techniques for Analyzing Surface Defects in Metal Castings", AI-TR-457, AI Laboratory, M.I.T., June 1978.

THE SRI IMAGE UNDERSTANDING PROGRAM *

M. A. Fischler (Principal Investigator)
Andrew J. Hanson (Testbed Coordinator)

SRI International, Menlo Park, California 94025

ABSTRACT

Our principal objective in this research program is to obtain solutions to fundamental problems in computer vision that have broad military relevance, particularly in the areas of cartography and photo interpretation. In addition to our own research, we are designing and implementing an integrated testbed system that incorporates the results of research produced throughout the image understanding community. This system will provide a coherent demonstration and evaluation of the accomplishments of ARPA's Image Understanding Program, thereby facilitating transfer of this technology to the appropriate military organizations.

I INTRODUCTION

Research at SRI International under the ARPA Image Understanding Program was initiated to investigate ways in which diverse sources of knowledge might be brought to bear on the problem of analyzing and interpreting aerial images. The initial, exploratory phase of research identified various means for exploiting knowledge in the processing of aerial photographs for such military applications as cartography, intelligence, weapon guidance, and targeting. A key concept is the use of a generalized digital map to guide the process of image analysis. The results of this earlier work were integrated into an interactive computer system called "Hawkeye" [1]. This system provides necessary basic facilities for a wide range of tasks in cartography and photo interpretation.

Research subsequently focused on development of a program capable of expert performance in a specific task domain--road monitoring. The primary objective of this continuing research has been to build a computer system, called the Road Expert, that "understands" the nature of roads and road events. It is now capable of performing such tasks as

* Finding roads in aerial imagery.

* Distinguishing vehicles on roads from shadows, signposts, road markings, etc.

* Comparing multiple images and symbolic information pertaining to the same road segment, and deciding whether significant changes have occurred.

The general approach, and technical details of the Road Expert's components are contained in References [2-7]. We have integrated these separate components into a coherent system that facilitates testing and evaluation, and have transferred this system to the ARPA/DMA Testbed.

In the most recent phase of our work, we have initiated major efforts in two new directions. The first is in support of a joint ARPA/DMA program to provide a framework for demonstrating and evaluating the applicability of image understanding research (from throughout the entire IU community) to military problems in general, and to the problems of automated cartography in particular. Our plans and progress in this effort (ARPA/DMA Testbed) are described in an appendix to this paper.

The goal of our second major effort is to broaden the scope and generality of our image understanding research--specifically in the areas of three-dimensional terrain understanding, perceptual reasoning, linear-feature analysis, and image description and matching. A parallel research program (described in Reference [7]), jointly supported by ARPA and NSF, complements these investigations by focusing on fundamental computational principles underlying the early stages of visual processing in both man and machine.

II RESEARCH PROGRESS AND ACCOMPLISHMENTS

Our current research efforts are focused on three somewhat independent problem domains:

* Detection, delineation, and interpretation of linear features in aerial imagery.

* The research described in this paper is based on work performed under Advanced Research Projects Agency Contract No. MDA903-79-C-0588.

* Image matching and image-to-Database correspondence techniques (including landmark selection and detection techniques for application to autonomous vehicle navigation).

* 3-D computation and interpretation (stereo matching, modeling, and raised-object cueing).

A. <u>Research in Linear-Feature Analysis</u>

The task of analyzing and interpreting linear features in aerial photography has a natural partition based on the resolution of the imagery.

In working with low-resolution images, where the linear structures are "line-like" in appearance, we have been primarily concerned with techniques for automatic delineation. A summary of our past work in this area is contained in Reference [6]. In that paper we described road domain applications of the techniques we had developed for combining and linking multisource information. We are now investigating the application of these techniques to other types of low-resolution linear structures. The key step here is finding suitable models to describe the "local" appearance of these linear structures. Figure 1 shows some preliminary results in the detection and delineation of rivers.

In high-resolution imagery, where the internal details of the linear structures are visible, the models and tracking techniques are necessarily somewhat domain-specific. A very successful approach to road tracking is described in Reference [3]. Our current work in this area addresses the problem of identifying and describing generic classes of objects that appear within the road boundaries (cars, shadows, road markings, etc.). Our approach is first to dynamically model the nominal road surface intensity pattern, next to use this model to reduce the background to a relatively homogeneous field in which anomalous objects (i.e., those that occlude the nominal road surface) are easily detected, and, finally, to inspect and classify the detected objects. Figure 2 shows an example of the steps in this process; a summary of the work is contained in Reference [9].

B. <u>Research in Image Matching</u>

Image matching is one of the broadest and most basic operations in scene analysis. Our current work in this area is concerned primarily with putting an image into correspondence with an existing data base. This is a requirement for almost all cartographic and "knowledge-based" image interpretation applications and is the essential step in scene-based autonomous navigation. A summary of our relevant contributions is contained in Reference [4] and [8].

In Reference [8] we introduced a new paradigm (RANSAC) for fitting a model to experimental data. RANSAC is capable of interpreting/smoothing data that contain a significant percentage of gross errors. It is thus ideally suited for applications in automated image analysis in which interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper [8] describes the application of RANSAC to the Location Determination Problem (LDP): given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. New results are derived for the minimum number of landmarks needed to obtain a solution, and algorithms are given for computing these minimum-landmark solutions in closed form. These results form the basis for an automatic system that can solve the LDP even under severe viewing and analysis conditions.

We have now begun an investigation of the problem of how to select and identify landmarks automatically--i.e., the design of a "Landmark Expert." This effort will benefit from our work on 3-D compilation and raised-object cueing.

C. <u>3-Dimensional Compilation and Interpretation</u>

We have initiated a number of new efforts in stereo matching and stereo modeling. With respect to the former, we are investigating the feasibility of an adaptive matching technique whose parameters are determined by the local geometric and photometric scene characteristics. As regards to stereo modeling, we are developing methods for analyzing areas that cannot be matched, as well as for identifying occlusion edges and small raised objects. As part of this new work, and also as an adjunct to the testbed effort, we have written a paper [10] that presents an overview and evaluation of the state of the art in stereo compilation.

Stereo Matching

The goal of a stereo compilation system is to generate a three-dimensional model of a scene, given two or more images that have been taken from different perspectives. There are at present two techniques for automatic stereo matching: correlation area matching and low-level feature matching. Each of these has both appropriate and inappropriate scene domains. Interestingly, many of the domains where correlation performs well are those in which feature matching performs poorly, and vice versa [10]. Furthermore, none of the current matching techniques makes use of the physical constraints that result from knowledge of the illumination source, the photometric properties of surfaces, and the geometric properties of natural and cultural objects. We are investigating new techniques that can smoothly combine several existing matching techniques to exploit available physical constraints and, thus attain levels of matching performance impossible with any single technique.

Major improvements of the feature-based approaches should result from increasing the number of semantic labels in the classification of edges to include shadows, occluding contours, and changes in orientation of a photometrically uniform surfaces, as well as from using the physical constraints between labels to guide the matching process (relaxation schemes will be used here). Similarly, improvements in area-based (correlation) approaches should result from combining stereo anomaly detection with the matching process to do a more effective job of geometrically shaping and warping the correlation windows (e.g., avoid having a correlation window overlap an occlusion edge).

An approach to the integration of feature- and area-based stereo techniques is to first decompose the images into regions according to local properties that affect the performance of the various matching techniques. For instance, regions that are photometrically uniform and have distinct boundaries are obvious choices for edge-based stereo techniques, whereas highly textured areas might be better suited to area-based methods.

Stereo Modeling

In real-world scenes, which contain such complex 3-D structures as groups of buildings, trees, and other elevated objects, there will always be portions of the imagery that cannot be matched because of occlusion, excessive perspective distortion, and temporal changes in the scene. These areas will be referred to as stereo matching anomalies. In many cases the existence of stereo matching anomalies can provide a cue to the presence of raised objects. Our research goal is to classify these stereo anomalies using supporting evidence from other cues, such as shadows and monocular features.

We are considering a number of possible approaches to raised-object cueing. The existence of a stereo anomaly might trigger a search to find a shadow consistent with the illumination geometry and the geometry of the underlying terrain. Alternatively, knowledge of the illumination geometry and the underlying terrain might be used to guide a highly selective, monocular search for shadow edges, which in turn would guide a search for anomalies in stereo matching. In addition to the above, we are also developing techniques for recognizing the presence of occlusion edges in single black and white images that could provide guidance for the stereo modeling system.

ACKNOWLEDGMENT

Contributors to the SRI Image Understanding Program include the following staff members: S. Barnard, R. C. Bolles, M. A. Fischler, T. D. Garvey, A. J. Hanson, D. L. Kashtan, L. H. Quam, A. P. Witkin, and H. C. Wolf.

REFERENCES

1. H. G. Barrow, et al., "Interactive Aids for Cartography and Photo Interpretation: Progress Report, October 1977," in Proceedings: Image Understanding Workshop, pp. 111-127 (October 1977).

2. M. A. Fischler, et al., "Interactive Aids for Cartography and Photo Interpretation," Semiannual Technical Report, SRI Project 5300, SRI International, Menlo Park, California (October 1978 and May 1979).

3. L. Quam, "Road Tracking and Anomaly Detection," in Proceedings: Image Understanding Workshop, pp. 51-55 (May 1978).

4. R. C. Bolles, et al., "The SRI Road Expert: Image-to-Database Correspondence," in Proceedings: Image Understanding Workshop, pp. 163-174 (November 1978).

5. G. J. Agin, "Knowledge-Based Detection and Classification of Vehicles and Other Objects in Aerial Road Images," in Proceedings: Image Understanding Workshop, pp. 66-71 (April 1979).

6. M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf, "Detection of Roads and Linear Structures in Aerial Imagery," in Proceedings: Image Understanding Workshop, (November 1979).

7. M. A. Fischler, "The SRI Image Understanding Program," in Proceedings: Image Understanding Workshop (November 1979).

8. M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," in Proceedings: Image Understanding Workshop (April 1980).

9. H. C. Wolf, M. A. Fischler, and L. Quam, "Road Domain Anomaly Analysis," (in preparation).

10. S. Barnard and M. A. Fischler, "An Image Understanding on Computational Stereo," in Proceedings: Image Understanding Workshop (April 1981).

APPENDIX

ARPA/DMA IMAGE UNDERSTANDING TESTBED PLAN

SUMMARY: We present a plan for the activities supporting the ARPA/DMA Image Understanding Testbed. A detailed schedule is given for the second half of FY81, together with an outline for the FY82 activities. The testbed efforts concentrate on three main areas: 1) the development of a software and hardware environment to support the requirements of the contributed software modules; 2) acquisition and integration of the contributions; and 3) demonstration, testing and evaluation of the image understanding research contributions.

TABLE OF CONTENTS

I OVERVIEW OF THE IMAGE UNDERSTANDING TESTBED

A. Background

ARPA has for many years been a primary sponsor of basic research in computer vision. This support was consolidated in 1975 into a broadly based research program in image understanding (IU), the goal of which was to explore fundamental computer vision techniques that could be applicable to military image interpretation tasks.

The IU research program has now produced a substantial body of results that have considerable potential for near-term application. To provide a framework for demonstrating some of these capabilities, ARPA and DMA have agreed jointly to support development of a demonstration system, the ARPA/DMA Image Understanding Testbed. While the immediate goal of the testbed is to explore applications of direct relevance to automated cartography, the ultimate results may be of interest to a far larger community.

B. Progress

SRI International was selected as the integrating contractor to implement the testbed system. Over this past year the SRI vision group has carried out the definition, planning, and facility implementation tasks of the testbed design. A basic configuration of hardware and software has been established that now forms the core of the testbed facility. In the following paragraphs we describe the hardware, system software, and language utilities currently available to support testbed activities.

1. Hardware Configuration

The principal elements of the current testbed hardware configuration are a DEC VAX-11/780 central processing unit and a DeAnza refreshed-raster- scan display system.

The DEC KL-10 central processing unit is also accessible from the testbed VAX, using a teletype line for communication and a shared disk drive for file transfer. Programs on the KL-10 can access the DeAnza display system through the VAX as an intermediary employing this same communication system.

The VAX is a four-megabyte system with one tape drive, two RP06 disk drives (one shared with the KL-10), and 16 teletype lines. The VAX interfaces directly with a variety of terminals, a digitizing table, a menu tablet, and a DEC PDP-11/34 minicomputer that controls the DeAnza display system directly.

The DeAnza refreshed-raster-scan display system has a resolution of 512 x 512, with 32 bits of information per pixel. Eight bits each are allocated to red, green, and blue data; in addition, there are eight overlay planes. Each group of eight bits has its own lookup table. The output of the four lookup tables can be combined in a variety of ways to provide input to the system's eight DACs. Ordinarily, three DACs drive one RGB monitor, three drive a second color monitor, and the remaining two DACs drive two monochrome monitors. However, a special crossbar arrangement has been designed and built to allow the DeAnza bit planes to be allocated dynamically among our two color monitors and up to eight monochrome monitors located with terminals throughout the site. All DeAnza graphics operations are carried out by the PDP-11/34 under the direction of the VAX.

2. Operating System Facilities

The testbed system runs under the UNIX operating system, which is currently available on the SRI VAX through an interfacing software package (developed at SRI) called "EUNICE."

The KL-10 presently associated with the testbed VAX runs under the TOPS-20 operating system; this facility is available to run application programs developed prior to the initiation of the testbed effort.

3. Language Support

High-level programming languages currently available on the testbed VAX are MAINSAIL, FRANZLISP, and C. Extensive graphics functions are currently available from MAINSAIL, which is a high-performance ALGOL-like language similar to the SAIL language available on PDP-10 computers ("MAINSAIL" is derived from "MAchine INdependent SAIL"). MAINSAIL is currently available on the PDP-10 under TOPS-20 and on the VAX under either VMS or UNIX. FRANZLISP, a variant of LISP, was developed at U.C. Berkeley; it is written almost entirely in C and is intended to run with the UNIX operating system (it runs under EUNICE on the testbed VAX). The testbed graphics capabilities are currently being extended to include C and FRANZLISP.

The languages MAINSAIL, SAIL, and MACLISP are supported on the KL-10. Graphics functions are accessible by using MAINSAIL on the KL-10 to send directives through the VAX.

C. Planned Activities

Activities designed to fulfill the goals of the Image Understanding Testbed project will fall into three major categories: development of system features and documentation; acquisition and integration of contributed image understanding modules; and the demonstration, testing and evaluation of the image understanding research contributions.

The testbed hardware and software configuration will be enhanced to establish a powerful, self-contained system for demonstrating the results of image understanding research. An effort will be made to configure the system so that similar systems could easily be duplicated at other sites. A full set of documentation describing the testbed configuration and its use will be generated. The documentation will include a user's manual, a description of the hardware and software of the testbed environment, a catalog of the supported contributions, and a catalog of the available test imagery. The documentation for the basic utility packages will establish community-wide standards for the image understanding software environment in C, FRANZLISP and MAINSAIL.

Image understanding modules will be proposed by each of the IU research groups and evaluated with respect to their suitability for the testbed. Those modules that are appropriate for a particular stage of development will be acquired. The support structure required by a given module will be analyzed and the module integrated into the testbed system.

The demonstration, testing, and evaluation procedures for contributed modules are currently being developed. As an initial step, we are undertaking a study of the fundamental functional areas and paradigms of IU research. The first such area being studied is stereographic systems. A paper on the state of the art of IU stereo modules is being prepared and will be presented at the April 1981 Image Understanding Workshop. As the testbed progresses, analyses of various IU areas and plans for appropriately demonstrating their capabilities will be developed regularly.

The testbed program will be implemented primarily by a team consisting of the testbed coordinator, the testbed system programmer, and a person with a vision research background who will play a major role in the integration of contributed modules. These personnel will be assisted by members of the SRI vision research group, as well as by consultants from testbed contributor sites. In particular, research personnel are expected to play a substantial role in formulating the test and evaluation plans and procedures for contributed modules.

II PROJECT PLAN FOR THE SECOND HALF OF FY81

A. Overview

Now that a basic testbed system concept has been established, the testbed must be evolved to accommodate the requirements of the software contributions anticipated from the other testbed participants. The proposed contributions must be examined and selections made. Consulting assistance by the original authors must be made available, if possible, to help identify the critical issues involved in adapting each contribution to the testbed environment. As the list of nominal contributions is developed, they will be brought into the testbed and integrated into its environment. The knowledge required to carry out the module integration task is still being acquired. Thus, the actual instantiation of the testbed will probably differ somewhat from the plans presented here.

While the contributions are being integrated, we shall also be evaluating the state of the art of in selected areas of image understanding. This activity will provide an essential basis both for the decisions to acquire specific modules and for the later task of evaluating the image understanding research results. The ultimate goal is to develop a comprehensive picture of the nature and capabilities of the major areas of the image understanding research program.

Table 1 contains a list of the major anticipated activities and milestones of the Testbed project.

Table 1

TESTBED ACTIVITIES AND MILESTONES

ACTIVITY	BEGIN	END
	(-- = ongoing activity)	
SYSTEM DEVELOPMENT		
<software>		
Operating System Environment	in progress	--
Graphics and Image Access Utilities	2/81	7/81
User Interface	4/81	1/82
<hardware>		
Image Scanner	12/80	3/81
Display	3/81	7/81
Lisp Machines	8/81	10/81
Other Hardware	3/81	11/81
<documentation>		
Plans and Presentations	11/80	--
System Environment	11/80	--
Graphics and Image Access Standards	6/81	8/81
User's Manual	8/81	10/81
Contribution and Image Catalog	6/81	--
ACQUISITION AND INTEGRATION OF CONTRIBUTIONS		
Definition	in progress	--
Evaluation and Selection	in progress	--
Acquisition	3/81	--
Testing and Modification	3/81	--
Integration	5/81	--
Test Image Acquisition	in progress	--
DEMONSTRATION, TESTING, AND EVALUATION OF CONTRIBUTIONS		
Develop Scientific Overview	1/81	4/82
Establish Test and Evaluation Plans	6/81	4/82
Demonstration of Contributions	6/81	--
Evaluation and Comparison Tasks	10/81	--

B. Development of System Software, Hardware and Documentation

The testbed system environment will be extended in a variety of ways during the remainder of this funding period. The purpose of these enhancements will be to provide support to facilitate evaluation of the contributed image understanding modules. The anticipated testbed system activities during the rest of FY81 may be grouped into three subcategories: software tasks, hardware tasks, and documentation tasks. These tasks will be planned by the testbed coordinator and carried out by all testbed personnel.

1. Software Development

OPERATING SYSTEM ENVIRONMENT

The IU modules contributed to the testbed will be written in a variety of languages and will be designed for several different environments. The testbed development plan is centered on a UNIX-based environment supporting the languages C, FRANZLISP, and MAINSAIL. A number of system software tasks will be undertaken to support the contributed modules. For example, we shall establish a set of basic software intercommunication capabilities; the objective is to make it very easy for programs written in different languages, such as C and FRANZLISP, to function cooperatively in an interactive environment. Communication facilities will also be established to allow interaction with other computer systems, such as the KL-10 and LISP Machines. Improvements in basic system utilities will be made as the need arises.

A software documentation system will be established that provides a framework for creating documentation in a standardized format. The documentation entered in this fashion will be entered into a database and we will be retrievable by a suitable documentation access system.

GRAPHICS AND IMAGE ACCESS UTILITIES

A standard set of graphics and image access utilities form a critical central part of the testbed environment. These are the tools that enable the rest of the IU modules to work together in an efficient way. Parallel sets of graphics utilities with essentially the same capabilities will be written in C, in FRANZLISP and in MAINSAIL to allow access to the graphics in all supported testbed languages. Image file access will be supported by parallel utilities in MAINSAIL, C, and FRANZLISP, so that image files can be accessed and manipulated in all of the supported testbed languages. The identification, labeling and retrieval of image data will be supported by an image retrieval database system. We are considering modeling the image database system on the HAWKEYE project previously undertaken at SRI.

USER INTERFACE

The testbed will support a standard interactive user interface to allow demonstration of and experimentation with all of the testbed capabilities. The user interface will be a powerful tool enabling exhaustive testing and evaluation of the contributed modules. The testbed user interface is currently envisioned as a LISP-based set of interactive utilities that can be invoked either in immediate execution mode or from a stored executive file. Prearranged demonstrations can be assembled in the form of a deferred execution file. Experimental evaluation can be carried out in the interactive mode. Where appropriate, keyboard command entry will be supplemented by the option of entering commands and command parameters by means of a pointing device, such as a digitizing tablet. For example, a tablet would naturally be used for pointing to delimiting boundaries inside an existing display while a user was defining a new, magnified display.

The contributions of image understanding research software would be organized into a documented database structure of their own. This system would contain within it predefined demonstrations of each module's capabilities.

2. Hardware Acquisition And Integration

Additional hardware will be acquired to enhance the present capabilities and to support all the requirements of the testbed. The testbed hardware will be integrated into the system, together with the required software support and utilities necessary to make use of it. Listed below are the major items of testbed hardware to be incorporated into the system during the coming year.

HIGH-RESOLUTION IMAGE SCANNER

The testbed will acquire and integrate a high-resolution optical scanner for the purposes of digitizing film images. The scanner's basic capabilities will be to digitize film images of up to 10 x 10 inches at a resolution approaching that of the film grain, approximately 10 microns. A high level of geometric precision will be provided. The photometric range will include 8 bits of monochrome and up to 8 bits each for red/green/blue scans of color imagery. The scanner will support a variety of functions. First, it will allow testbed visitors to bring their own test data in the form of film imagery, digitize the data on the spot, and test any given module on their own data. Second, the scanner will provide a vital opportunity for experimenting with imagery digitized under controlled conditions; testbed algorithms can then be evaluated with respect to their sensitivity to a variety of factors in the image digitization process. Finally, the scanner will support the geometric precision and photometric accuracy required to carry out stereo mapping functions, shadow identification and raised-object identification; previous vidicon-based digitizing systems have proven completely inadequate for these purposes.

GRAPHICS DISPLAY SYSTEM

A refreshed-raster-scan graphics display system compatible with a majority of the testbed contributors systems will be added to the testbed system. The graphics system will extend the capabilities of the testbed graphics, so that existing contributed software can be integrated more easily into the system. Since the current testbed graphics system is a specially modified research system developed at SRI, it will be highly beneficial to the testbed to have a standardized system of its own that can be easily duplicated and maintained. One of the tentative goals of the testbed project is to develop a transportable and universally duplicable system with capabilities adequate for all image understanding tasks. The enhanced graphics system plays a critical role in allowing the testbed configuration to meet these goals.

LISP MACHINES

A number of highly desirable image understanding contributions are coded in LISP-Machine LISP and run properly only on the MIT "CADR" LISP machines. To take full advantage of the capabilities of the MIT contributions and to accommodate later contributions that may be coded in LISP-Machine LISP, it would be highly desirable to install at least two LISP Machines as part of the testbed hardware configuration. These machines would operate as independent processors for intensive LISP calculations, thus relieving the VAX of the substantial processing load required to service such computations. The LISP machines would communicate with the VAX via the Ethernet high-bandwidth network, using existing software that is available for the LISP Machines.

OTHER HARDWARE

The testbed will acquire and integrate three new high-capacity disk drives. The additional disk drives will be used to replace the shared drive that is being lost and to increase the system's capacity to support a substantial number of users and their test imagery.

A high-quality film hard-copy device for photographing intermediate and final results of image understanding computations would be a highly desirable addition to the testbed hardware system. Consequently, it will be added to the system whenever appropriate funding is available.

Communication with other computer systems and peripherals will be essential to satisfactory operation of the system. We have already begun installing an Ethernet network on the Unibus of the testbed VAX. High-speed communication channels will be established using the Ethernet to communicate with other VAXs, the KL-10, the 2060, and the LISP Machines.

3. Documentation

The establishment of a full set of documentation describing all aspects of the image understanding testbed system will be a critical part of the testbed project. Among the specific documentation tasks that must be included in the testbed plan are the generation of testbed proposals, plans, and reports, the preparation of informational presentations, descriptions of the testbed hardware and software environment, and descriptions of both the scientific and user-related properties of the contributed testbed modules.

PROJECT DOCUMENTATION-----------------------------

PROJECT PLANS

Project plans detailing the anticipated development of the testbed will be prepared as required. The plan presented in this document is the first detailed plan attempted for the testbed project as a whole. As the testbed evolves and changes, the project plan will be updated accordingly.

PROJECT PRESENTATIONS

Periodically the status of the testbed must be formally presented to such groups as the Testbed Steering Committee and the IU principal investigators. These project presentations will be developed as required to communicate information regarding testbed activities to concerned parties.

SYSTEM DOCUMENTATION------------------------------

HARDWARE ENVIRONMENT

A succinct but comprehensive description of the present testbed hardware configuration is now available. This document will be expanded and updated as the testbed hardware configuration and critical information about the configuration are changed. Further diagrams and details of the system hardware usage will be added to improve the utility of the existing descriptions.

SOFTWARE DEVELOPMENT ENVIRONMENT

A brief description of the present testbed system software development environment is currently available. As additional utilities and support software become available or are generated for the testbed, a more comprehensive documentation scheme for the testbed will be established. An indexed set of documentation describing the basic software characteristics of the testbed environment will be generated and periodically updated when appropriate.

GRAPHICS STANDARDS

All graphics manipulation functions will be available with three alternative environments - MAINSAIL, C, and FRANZLISP. A set of documents describing the available testbed graphics functions, their parameters, and their usage will be generated as an integral part of the testbed system. The MAINSAIL documentation is in progress, while the C and FRANZLISP documentation will be generated concurrently with the relevant utility software.

So far as is practical, a universal set of testbed graphics standards will be created that all contributions will use for graphics access. The testbed graphics function documentation will establish this de facto standard.

IMAGE ACCESS STANDARDS

All image access functions will be available in three alternative environments - MAINSAIL, C, and FRANZLISP. A set of documents describing the available testbed image access utilities, their parameters, and their usage will be generated as an integral part of the testbed system. The MAINSAIL documentation is in progress, while the C and FRANZLISP documentation will be generated concurrently with the software.

So far as is practical, a universal set of testbed image access standards will be created that all contributions will use for image access. The testbed image function documentation will establish this de facto standard.

USER'S MANUAL

The testbed user interface system will be described in a detailed testbed system user's manual. The user's manual will contain a brief overview of the testbed system hardware and software configuration, together with instructions for using the entire system from within the user interface shell. The methods and utilities available for carrying out testing and evaluation procedures will be described in detail. Basic instructions for using the program development environment will be supplied together with pointers to more complete documentation. The user's manual will also indicate how to use the on-line documentation to obtain more information about specific contributions and their usage within the testbed context.

TESTBED CATALOGS----------------------------------

CONTRIBUTION CATALOG

The capabilities and image understanding research modules available on the testbed will be fully documented. A contribution catalog will be established on line to allow quick access to essential documentation describing the use of the contributed modules. Each contribution will be described both in terms of its scientific content and its practical usage. A series of pointers will be set up among related routines, as well as among those that can be used cooperatively in some fashion.

TEST IMAGE CATALOG

The test image catalog will be an essential part of the testbed system's on-line documentation. It will contain both human-readable pointers to test data falling into various categories and machine-readable pointers that enable the user to select particular types and instances of test data with minimum knowledge about the explicit file structures involved. The test image catalog will work in concert with the test image database system to form a total environment that is tentatively based on the SRI Hawkeye image database system.

C. <u>Acquisition and Integration of Contributions</u>

A major portion of the testbed effort over the next year and a half will be devoted to acquiring image understanding contributions from the other testbed contributors at Carnegie-Mellon, MIT, Stanford, Rochester, Maryland, and the

University of Southern California. After the acquired modules have been integrated into the testbed system, they will be tested, demonstrated, and evaluated to weigh their technical merits. The sequence of procedures we plan to follow while adding contributed modules to the testbed system is listed below. The acquisition and integration of contributed modules will be planned by the testbed coordinator and carried out by testbed personnel with substantial assistance from home site consultants.

DEFINITION

Contributors have been asked to describe their proposed modules in detail. This information should to give testbed planners a sufficiently accurate picture of the capabilities and degree of completeness of the proposed modules.

SELECTION

Proposed modules that both the contributing organization and the testbed staff consider potentially suitable for inclusion in the testbed will be evaluated more intensively. Most or all such modules will be demonstrated at their home site to determine their capabilities, degree of portability, and desirability. Possible problems in transferring the modules to the testbed environment will be discussed in detail. Qualifying modules will then be selected for inclusion in the testbed system.

The qualification and selection procedure will be carried out in stages. Small, simple modules will be preferred at first. As the testbed software enviroment is enhanced to accommodate the requirements of the initial modules, it will become easier to support more complex contributions. In addition, modules that were still incompletely implemented during the early stages of the module acquisition process will gradually be completed. As this is accomplished, additional contributions will be qualified for inclusion in the testbed.

ACQUISITION

Modules that are selected for inclusion in the testbed at any given stage will be prepared by the contributor for transport to the testbed. In many cases, this preparation will be sufficiently difficult that software consultants familiar with the contribution and its structure may visit the testbed site in person to assist in making that contribution operational. Acquisition of a given contributed module will include as much support software as is practical.

TEST AND MODIFY

Acquired image understanding modules will be tested and modified to fit into the testbed environment. When necessary, consultants from the respective module development sites will be employed to carry out this procedure efficiently. This activity is the essential prerequisite to integrating the contribution into the testbed system.

INTEGRATION

When a contributed module is sufficiently well understood so that its entire support structure has been properly implemented in the testbed environment, the module can be integrated into the testbed system itself. At this stage each contribution will be added to the list of processes that can be interactively invoked and experimented with on the testbed. Appropriate ways of passing data to the module from the interactive environment will be established and documented and, when desirable, interfaces to other modules will be put in place. One or more test cases will be set up to demonstrate the capabilities of each module.

D. <u>Demonstration, Testing, and Evaluation of Contributions</u>

The primary scientific task of the testbed is the evaluation of the contributed image understanding research modules in a uniform context. This task will be broken down into a number of distinct stages. SRI testbed and research staff will cooperate in developing a scientific overview of the state of the art of image understanding research. From this work will come a more detailed plan for testing and evaluating the contributed modules. The actual demonstration, testing, and evaluation of the image understanding research contributions will be based on that plan and will be carried out primarily by the testbed personnel.

SCIENTIFIC OVERVIEW

A scientific overview of the state of the art of image understanding research will be developed with the participation of image understanding researchers from the community of testbed contributors. The basic IU paradigms will be identified, the nature of the IU research contributions to each area described, and appropriate test domains noted. This process will result in a series of scientific papers reviewing and evaluating the status of the major IU functional areas. The following areas have tentatively been identified for this activity:

* Stereographic reconstruction
* Linear feature analysis
* Image matching
* Pattern recognition and segmentation
* Miscellaneous (including sensor prediction, texture analysis, shape from shading, etc.)

ESTABLISHING THE TEST AND EVALUATION PLAN

As the scientific overview is developed, we shall formulate a strategy for testing and evaluating the IU contributions in each of the major technical areas. A plan will be proposed for carrying out the evaluation procedure. Testing methods will be proposed and specific evaluation measures suggested. Appropriate domains of test data will be determined for each of the functional areas, as well as for the corresponding contributions.

DEMONSTRATION OF CONTRIBUTIONS

When a contributed image understanding module has been fully integrated into the testbed system, it will be added to the set of testbed demonstration packages. Facilities will be established to use the module not only with special test cases, but also with any set of test data the user wishes to try. Home site consultants may again be used at this time to optimize the quality of the demonstrations.

EVALUATION AND COMPARISON

The final step in the testbed program will be to evaluate completely integrated image understanding modules according to the test and evaluation plan. When possible, similar modules will be compared with one another. Further work will be done to determine which measures are appropriate for the evaluation of particular modules, which image domains are appropriate, and which imagery is representative of these domains. All conclusions and results of the evaluation procedure will be fully documented.

III PROJECT PLAN FOR FISCAL YEAR 1982

The major testbed system tasks during the second half of FY81 involve improvements in the testbed environment, integration of contributed software, and the establishment of a plan for evaluating contributed modules. Additional hardware is scheduled to be incorporated into the system and the software utilities will be substantially extended. An initial set of image understanding research contributions will be acquired and the testbed environment prepared to support them as they are integrated into the system. The state of the art of image understanding research will be determined and a coordinated plan for testing and evaluating the contributed modules will be formulated.

The specific tasks to be carried out in FY82 depend largely on the manner in which the project develops during the remainder of FY81. However, many of the activities anticipated for FY82 will clearly be extensions of those begun earlier. Tasks started in FY81 will be further defined and brought to completion in FY82. Segments of the system will be redesigned and improved when necessary. Obviously, the system requirements will have to be redefined as the testbed staff and users gain experience with the system.

The general thrust of testbed activities in FY82 will be to acquire, integrate, and evaluate a substantial additional number of contributed image understanding research results and to complete the testbed contribution evaluation program. As the library of image understanding software supported by the testbed grows, the scientific overview and the evaluation plan will be extended and modified. Additional scientific reports will be generated to evaluate the state of the art of the image understanding field and to point out areas of strength and weakness. These results will help to clarify the particular modules that should be supported by the testbed, as well as to pinpoint areas in which additional work and research would be warranted.

The tasks involved in the demonstration, testing, and evaluation of contributed modules (described in the preceding section) will be continued and extended. In particular, strategies will be developed for demonstrating the capabilities of each module and the modules will be tested and evaluated with respect to their relative performance and scientific merits. The principal contents of the final report relating to testbed activities will be a detailed description of the results of the evaluation procedure.

IV STATUS AND PLANS FOR SPECIFIC CONTRIBUTIONS

The purpose of this section is to present a summary of the specific image understanding research contributions that are expected to become a part of the testbed system. Each contributor has provided the testbed staff with a list of proposed modules and a description of their characteristics. These specific descriptions of what each contributor could provide have been collected in a computer file for reference. While the contents of this lengthy file will not be included here, we plan to compile an appropriately edited version containing a full description of each contribution actually received by the testbed.

Each contributor has been contacted and asked to furnish information regarding the most appropriate initial contribution from that contributor's site. The proposed contributions are being evaluated and are being prepared by the contributors for transport to the testbed. The contributors have also provided information on the contributions they anticipate having ready before the end of FY82. In Table 2 we summarize the presently planned contributions to the testbed from each participant. The degree of the contributor's commitment to providing each given module is indicated, along with approximate dates for the prospective acquisition of the module and its integration into the testbed system.

The preferable form of the delivered contributions would be in one of the languages supported on the testbed VAX. Initial contributions will be analyzed from the standpoint of their support requirements and an appropriate set of graphics and image access standards will be formulated. While several attempts to define testbed standards have already been attempted, the

Table 2

SUMMARY OF CONTRIBUTIONS AND CONTRIBUTION SCHEDULE

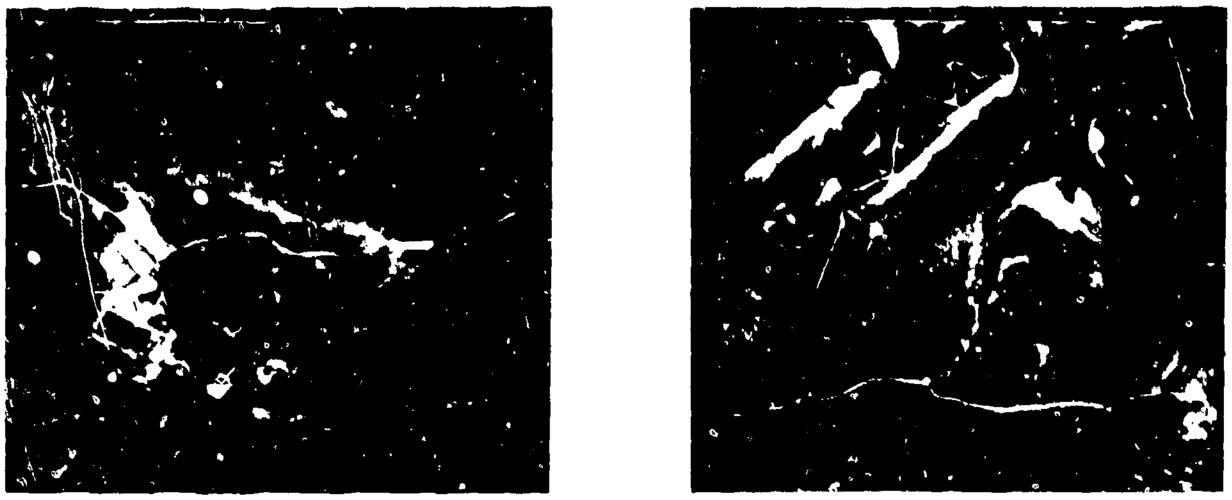
CONTRIBUTION	COMMITMENT	ACQUISITION	INTEGRATION
SRI			
Road Expert	firm	complete	5/81
RANSAC	firm	complete	5/81
CMU		(1st:2nd version)	(1st:2nd version)
Segmentation	firm	4/81:9/81	6/81:12/81
Intervisibility	likely	6/81:9/81	9/81:12/81
Stereo (Moravec)	likely	8/81:12/81	10/81:3/82
STANFORD			
[Stereo - see CMU]	[firm]	[8/81]	[3/82]
Camera Solver and 3D Obstacle Finder (Gennery)	likely	6/81	10/81
Line Finder	likely	6/81	11/81
ACRONYM	likely	4/82	6/82
MARYLAND			
Relaxation	firm	3/81	5/81
Interactive Segmentation Package	possible	9/81	12/81
ROCHESTER			
Hough Transform	firm	6/81	8/81
Strip Trees	possible	10/81	12/81
MIT			
[Require MIT LISP Machine]	likely	8/81	10/81
Stereo (Marr)	"	"	"
Shape-from-Shading	"	"	"
USC			
Linear Features	likely	available in SAIL	9/81
Law's Texture Analysis	likely	available in SAIL	12/81
Image-to-Map Correspondence	likely	available in SAIL	4/82

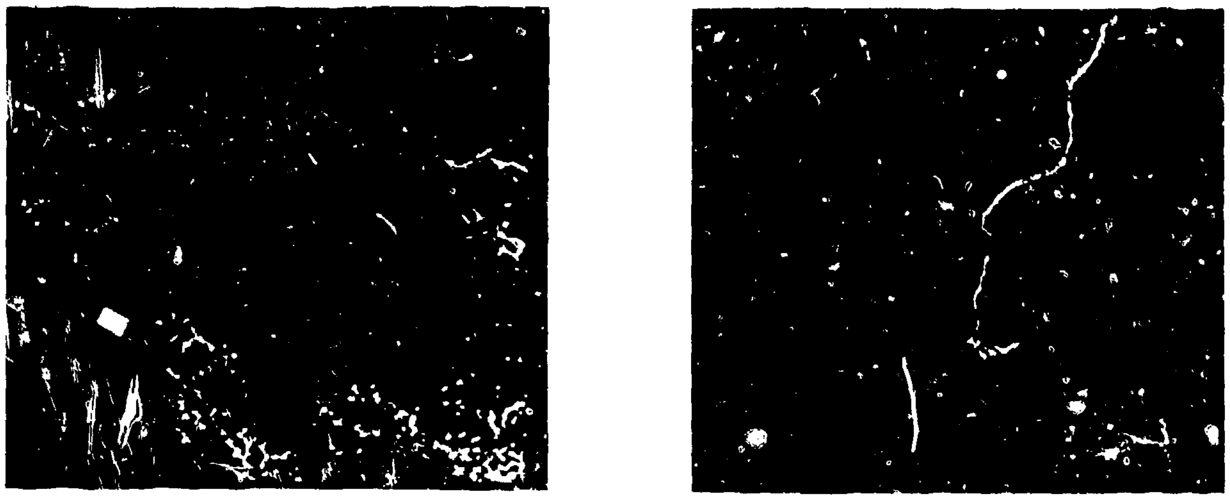
severe difficulties encountered in establishing ideal standards have indicated that they are best developed in an iterative fashion on the basis of existing software. Thus, the initial contributions will help define the standards to be followed in later contributions and in later versions of the initial contributions. We shall depend upon some support from each contributing institution to aid in fitting the contributions into a uniform environment.

The scientific context of each of the contributed modules will be defined by the scientific overview papers mentioned in Section II-D. The evaluation procedures and criteria to be used in the scientific assessment of each module's capabilities will be derived from the basic material in those papers. The appropriate data sets to be used in testing the contributions will also be characterized in the overviews. The final output of the testbed will be a systematic evaluation and characterization of the merits of each contribution, together with an environment in which the contributions can be flexibly tested and demonstrated.

VII CONCLUDING REMARKS

The image understanding testbed will establish a standard environment in which the image understanding research modules can be tested and evaluated. A variety of IU research results, implemented in the testbed system, will be accessible for evaluation and comparison. Once in place, the testbed will form a framework which other concerned parties, such as the Defense Mapping Agency, can use to adapt image understanding research capabilities to their own specific applications.


(a) ORIGINAL IMAGES


(b) BRIGHT POINTS IN BINARY MASK SHOW WHERE IMAGES HAVE "RIVER-LIKE" APPEARANCE

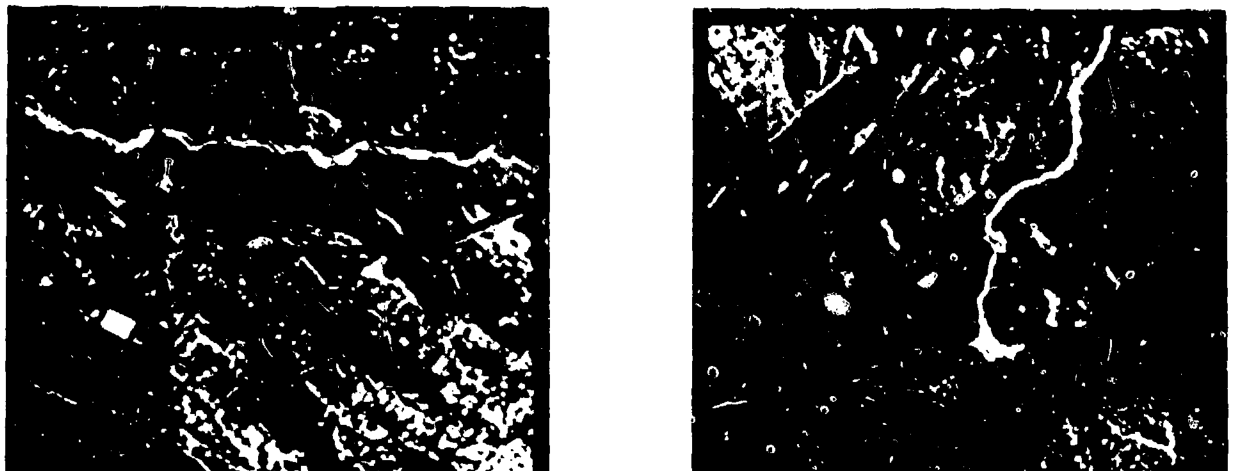
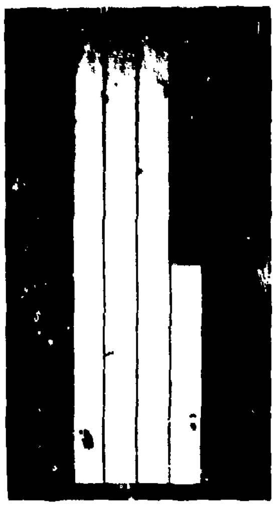
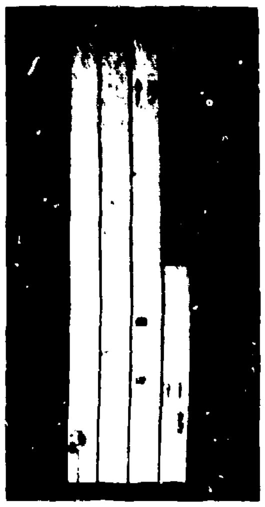
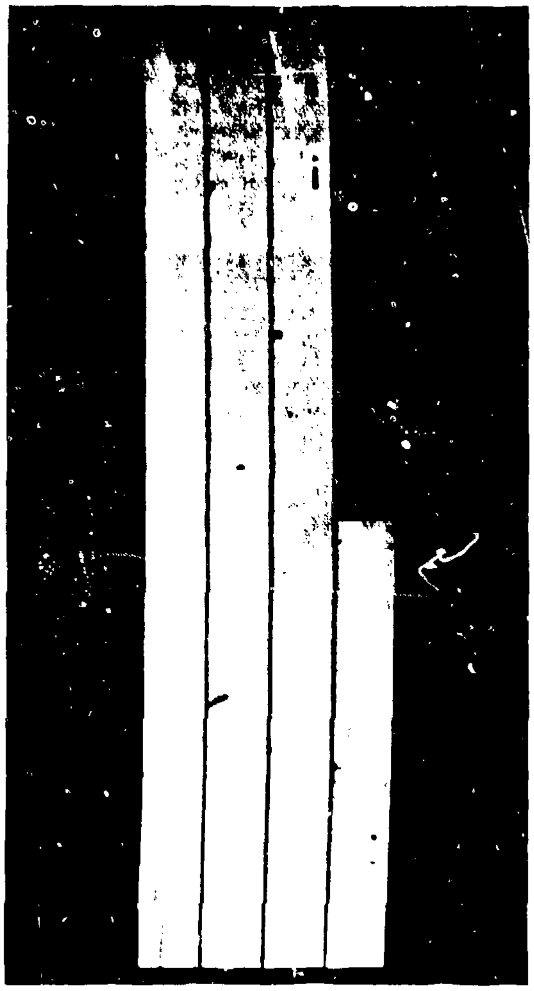

(c) RIVER DETECTION MASK OVERLAID ON ORIGINAL IMAGES

FIGURE 1 DETECTING AND DELINEATING RIVERS IN AERIAL IMAGERY

(a) ROAD SEGMENT TO BE MONITORED MARKED IN BLACK


(b) STRAIGHTENED, INTENSITY NORMALIZED ROAD SEGMENT


(c) DETECTED ANOMALIES MARKED IN BLACK


(d) BOXES PLACED AROUND POTENTIAL VEHICLES

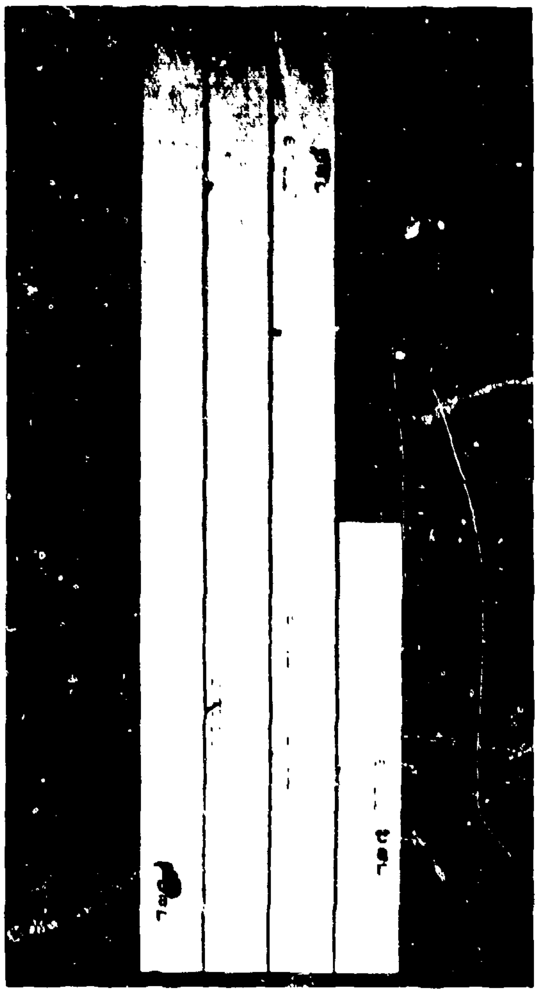

(e) ANOMALIES CLASSIFIED (LABELED)

FIGURE 2 ROAD DOMAIN ANOMALY DETECTION AND CLASSIFICATION

SPATIAL UNDERSTANDING

Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department
Stanford University, Stanford, California 94305

Abstract

The ACRONYM system has been extended greatly over the past year. It proved successful in preliminary tests in identifying aircraft. ACRONYM is domain-independent; its knowledge of aircraft is in the form of models. None of its rules and perceptual mechanisms are specialized to aircraft.

Progress has been made in stereo vision in geometric constraints for general surfaces and for special surfaces. General new theoretical results on understanding line drawings in images appear applicable to solids with curved surfaces with surface markings and shadows, with paper, and wires. In an example, a three-space model is built up of an aircraft from a single view, using general constraints with no knowledge of the object or surfaces. The system identifies shadows and uses them to infer the height of surfaces. Several new constraints on correspondence of edges have been discovered and incorporated in stereo matching. Two results relevant to biological vision systems are found to agree with experiment.

Concerning special surfaces, preliminary results are reported for monocular constraints on appearance of orthogonal trihedral vertices in image sequences.

Introduction

Our research program focusses on the ACRONYM system for perception and planning action in the real world. ACRONYM integrates perceptual algorithms in a total system which interprets images as spatial structures in three-space. Our interest in building a total system is to solve fundamental problems of representing geometric structures and transformations. The effort in systems is worthwhile, too, in providing an experimental system to build on for further research. An essential part of the research couples ACRONYM to real world inputs, by developing fundamental algorithms for constructing symbolic descriptions of images and sequences of images, and mapping them to symbolic descriptions of surfaces and objects.

ACRONYM is the basis for performance systems in photointerpretation and cartography, systems which promise to be generalizable from a few specially chosen examples in a research environment to realistic variation in an operational setting. Now, based on recent progress, we see perceptual components and an integrated system whose performance within the short term will be striking by current standards. Defense applications of image understanding are really quite difficult and demand that level of performance: automating stereo compilation for building complexes and cultural sites; automating classification tasks of photointerpreters; guidance and targeting. We set a short term objective of demonstrating interesting performance in example tasks, performance adequate to see a clear path to exploit image understanding technology for carefully selected high performance systems

ACRONYM

ACRONYM is a perceptual system with geometric modeling and geometric reasoning in the form of a powerful problem-solving system [Brooks 79]. Several objectives have influenced the design of ACRONYM. ACRONYM should be natural to program; it is programmed from geometric models. ACRONYM should incorporate all available knowledge and information; it maps knowledge and data into geometric constraints on various geometric structures of a geometric representation hierarchy. ACRONYM should be capable of perception without knowledge of viewpoint; it has viewpoint-independent volume models.

In ACRONYM, objects are represented as part-whole graphs whose primitives are generalized cylinders. One objective of research is to provide means for representing object classes and identifying them. Thus, ACRONYM has models for passenger aircraft and models for L-1011 and 747 as aircraft. It has succeeded in identifying 707's as aircraft without a model for the 707. Typical systems identify an aircraft by attempting separate identifications as an L1011, 747, or 707, without the least connection among them. In those systems, an L1011 is no more and no less related to a 747 than to a car or tank. Generalized cylinders provide a means of generic representation (object classes) as abstract structures of volumes. ACRONYM provides class restriction and quantifiers as further mechanisms for generic representation. A quantifier is a variable whose value is partially determined by a system of constraints. An L1011 is a restriction of the class passenger aircraft.

[Brooks 79] described a first test of ACRONYM in identifying an L-1011 from an aerial photograph of San Francisco airport. After a few such experiments, extensions to ACRONYM were designed and implemented over the course of more than a year [Brooks 80]. These extension introduced the major capability to extract three dimensional information from images. [Brooks 81] describes the first series of experiments testing the new system. ACRONYM performed well with poor data from small images. At this point, the major limitation is the quality of the symbolic descriptions it receives from edge and ribbon finding programs. We used to say that even though edge finders are weak, we can still see a lot in those images, "why can't computers?". That comment is no longer valid. In this case, ACRONYM understood poor data better than its author. Of course, he could do better than ACRONYM with the original image, instead of the faulty edge data. ACRONYM decided from shape that one aircraft was consistent with L-1011 and two aircraft in the example were not 747s or L-1011s. It did not use size, although ACRONYM concluded separately that they were too small to be 747s if the other were an L-1011.

ACRONYM had models of aircraft, a camera model with elevation between 1000 and 12000 meters. It had very little information about expected apparent size. ACRONYM has no special knowledge of aerial scenes. All its rules are about geometry and algebraic manipulation. Its performance would be impressive if it were a special program devoted to aircraft. It is more impressive that its mechanisms have nothing to do with aircraft, only its models. ACRONYM can be expected to perform well in identifying objects such as vehicles by substituting models for vehicles.

Extensions to ACRONYM included: generalization of the rule language to simplify the rules and to allow rules to manipulate object graphs and prediction graphs, in order to make control uniform; introduction of quantifiers, as described above; introduction of constraint expressions involving variables; symbolic manipulation and simplification of geometric transforms; incorporation and extension of a mechanism to test satisfiability of constraints; constraints as mechanisms for determining parameters of models from observations (also [Lowe 80]); prediction with parameterized expressions; a rule-based matcher.

ACRONYM's approach in identifying aircraft is to make predictions about the appearance of aircraft, i.e. to predict that on a coarse level, fuselage and wings will be observable. It predicts automatically image features and relations which are invariant or quasi-invariant over variations in the model and camera parameters. ACRONYM predicts shape as *ribbons* and ellipses. ACRONYM's data are ribbons found by a ribbon finder [Brooks 79b] based on output from the Nevatia and Babu edge finder [Nevatia 78]. ACRONYM uses its predictions to analyze ribbons to select candidates for aircraft. Matches of image features to model features are represented by constraints. A constraint satisfiability test removes inconsistent interpretations. Constraints on image measurements partially determine model parameters. The determination of model parameters allows a further prediction of fine detail, for example, prediction of engine pods and horizontal stabilizers based on the identification and location of an L-1011. Those predictions can be tested in order to verify identification to much higher detail.

Image Description

Performance of ACRONYM and stereo programs are both dependent on quality of symbolic descriptions of images as curves and ribbons. In the near future, significantly improved curve descriptions are expected. [Binford 81] describes new theoretical results on describing image boundaries, based on extension of the Binford-Horn line finder [Horn 72]. These results have not been implemented yet. An argument is made that coarse-to-fine operation is not useful; that fine-to-coarse is preferable because the output of fine operators can be used to eliminate the effect of small detail from coarse operators in a way that simple weighted averages (frequency filtering) cannot do. A conjecture was made about human performance on discriminating darker-vs-lighter for two background areas with readily discriminable fine texture. The contrast between the backgrounds is chosen to be small so that the discrimination is possible only over large areas. The contrast and density of small features is chosen so that their average intensity reverses the contrast between the two regions. The apparent result from a casual experiment is that humans detect the difference between backgrounds when the small features are discernable, and perceive the difference in the opposite sense when the small features are not discernable. Allan Miller and Craig Rublee synthesized the test images. The conclusion from this experiment is that position-invariant operators cannot be used at the coarse level to explain human performance, and should not be used by machines. The operators introduced in [Marr 77] are thus not adequate to explain human performance. That paper describes a solution to directional boundary operators

[MacVicar-Whelan 81] describes a boundary finder with sub-pixel accuracy. This is intended as a processor with intermediate performance requiring small computation. It is based on an approach similar to Binford-Horn [Horn 72] and [Marr 79].

[Miller 81] describes a VLSI implementation of a vision processor. This is an initial step toward high performance computation of image features such as boundaries.

Recent results suggest renewed support for the following design of a vision system: determination of image features with operators over a range of sizes; the output of each size stage is used to remove boundaries for the image data input to the next coarser stage; geometric grouping operations are performed on outputs of each stage, including the finest; relatively general, local assumptions are used to infer edges of surfaces from image boundaries. This model was unpopular in 1970. It appears a stronger candidate now because boundary operators and grouping operators use similar mechanisms which appear promising for implementation in VLSI, and because of the introduction of a general approach to interpretation of image boundaries [Binford 81].

Stereo Compilation

The chief problem of stereo vision is to find correspondences between areas and edges in one view, and those in another view. There is only partial correspondence between the two views which differ because of geometric differences, photometric differences, and physical changes from one photo to the next. Consider geometric image correspondence, the map between two images, which involves stretching, folding, and cutting, corresponding to surface inclination, discontinuities in tangent plane, and occlusion.

Much work has been directed toward correspondence along epipolar lines in an image [Henderson 79]. Arnold and Binford show that there is a tight constraint on corresponding edges [Arnold 80]. For edges randomly distributed in direction, most edges will appear to have similar angles in two views. This constraint is restrictive for typical mapping sequences of aerial photos, it is very tight for human stereo. Half width at half maximum for true matches is 9 degrees for human stereo at 1 foot. These results have consequences for biological stereo vision which are consistent with experiment. [Nelson 77] report half width at half maximum of 10-20 degrees for the cat. If randomly distributed edges are mismatched, i.e. different views correspond to different edges, they correspond to a population of edges peaked along the lines of sight to the two cameras.

The constraint on edge angles relates isolated edges without relations between edges. The next level of constraint is that of surfaces. An area in an image corresponds to a surface in space. Consider the interval between pairs of edges along an epipolar line, corresponding to a slice through the surface. For surfaces randomly oriented in space, projected intervals are nearly equal for most surfaces. Randomly mismatched intervals correspond to surfaces which are peaked along the lines of sights from the two viewpoints. Again, the constraint is useful for mapping sequences with wide stereo baseline, and very tight for human stereo which is small angle stereo.

For modeling expected edge and surface distributions, the model of a uniform distribution on the unit sphere (Gaussian sphere) is modified by superimposing a peak along the vertical and a band along the horizontal plane for cultural objects and the ground surface. Vertical edges in aerial photographs typically appear short; measurement of their direction is feasible with sophisticated edge operators.

[Clarkson 81] has found a version of the solution to the camera transform given only pairs of conjugate image points without spatial location. The method involves solution of a 3 parameter problem instead of a five parameter problem, thus it presumably requires much less computation. The solution must be within about .3 radians in initial estimates of orientation parameters. The objective of this work was to find a solution which is more robust than that of Gennery given noise points [Gennery 79]. This goal has not been achieved yet.

Interpretation

New theoretical results outline a theory of interpretation of line drawings which promises to include curved surfaces with surface markings and shadows, paper (non-solids), and wires [Binford 81]. These results appear to have definite application for stereo and photointerpretation. There appears to be a rich body of results to be obtained by extending that line of research. In particular, it is planned to extend the results to stereo coincidence. The approach begins by characterizing surfaces by their edges and limbs, i.e. apparent edges. It uses assumptions of general source position and general observer position to identify evidence that certain edges and surfaces intersect in space, and whether surfaces are smooth at apparent edges. Where curves are smooth at junctions, assume that the surface is smooth. Where curves have breaks, assume that the surface has a crease, i.e. discontinuous tangent plane. In absence of other information, if curves appear to intersect in an image, and if their inverse image is sufficiently constrained, assume that they are the images of curves which intersect in space. The inverse image of a curve in an image is the set of rays in space (a surface) which project onto the curve. If surface markings are drawings of objects, they will fool these interpretations, but when shadows and surface markings aren't constructed perversely, they give no indication of solid objects. Where shadows and surface markings cross true edges, the true edge is unaffected, the image of the edge is smooth. Thus the system infers that the surface is smooth along the shadow or surface mark. These results are much more general that previous theoretical studies of interpretation of line drawings [Guzman 68, Huffman 71, Clowes 71, Waltz 72, Mackworth 73, Turner 74].

Where shadow information is available, it enables accurate measurement of three-space positions of many edges in the scene, from which other dimensions can be inferred. Shadows provide information equivalent to stereo. Use of shadow information is an important capability for geometric reconstruction for photointerpretation. Use of shadows will be discussed below.

[Lowe 81] exploit some of these assumptions in interpreting an aerial photograph of an aircraft. Figure 1a shows image curves extracted by hand from output of the Nevatia and Babu line finder [Nevatia 78]. This is a cheat which represents a judgment of what line finders will produce in a few months. Figure 1b shows occlusion cues which provide a layered relative depth model: the fuselage is above the wing which is above the engine pod which is above the shadow which is on the ground; the shadow is over a white line which must be on a smooth surface. The techniques are quite general, they do not depend at all on knowing the objects in question, and they use cues which are universally available. Figure 1c shows matching of vertices of shadow images coincident with other junctions along a projection from the sun (coincidence assumption). The system determines heights of edges which cast shadows from triangulation with the known sun position. Figure 1d shows the resulting three-space model of the surfaces of the aircraft seen from several views.

There is much more still to be extracted from these curve data.

[Liebes 81] has developed a general formulation of the principles of the perspective geometry of shadow formation for local and remote point and extended sources, and has applied this formulation to a variety of basic geometrical configurations.

It is proposed to follow this investigation of shadows to some further results which appear to be directly ahead.

These monocular interpretations are not a substitute for stereo, but they can aid greatly in stereo correspondence. One problem with stereo systems has been that they don't use shape as humans do and they don't use context. There are at least three classes of monocular cues described above: first depth ordering of surfaces from occlusion cues, combined with object segmentation; second, shape description in terms of generalized cylinders; and third, quantitative depth models obtained from analysis of shadows. Correspondence is enormously simplified if there is a qualitative ordering of surfaces, and simplified more still if there is a quantitative model. That is, matching top surfaces with top surfaces, and ground surfaces with ground surfaces leaves small search spaces.

Orthogonal Trihedral Vertices

[Liebes 81] has achieved a number of preliminary results on the use of geometric constraints for special surfaces in stereo matching for orthogonal trihedral vertices. These provide monocular cues for correspondence.

The objective of determining a depth map $z(x,y)$ can be aided by symbolic description of portions of the scene. In some portions of an image, little or no depth information can be obtained, e.g. on water, snow, concrete, gravel roofs, on uniform metal surfaces, etc. The best information about surface height in such cases can be obtained by fitting a surface satisfying boundary conditions with symbolic constraints. A lake is an extreme case. The condition applies to the height of the water at the lake boundaries with the constraint that the water surface is horizontal. On a horizontal circular cylinder, the surface has boundary conditions that the position at the boundaries of visible surface is known and the the surface tangent is known to be along the line of sight. Thus, symbolic constraints provide ways of translating familiarity cues into improved measurements.

The direction of the gravity vector is perhaps the single most important determinant in the alignment of architectural structures. Walls of buildings and their edges tend to be vertical. Most of the visible area in an aerial photograph of a building is the roof. Roofs are usually composed of planar sections, many horizontal. Architectural structures contain many right angles and orthogonal trihedral vertices, at intersections of walls and roofs of concrete buildings, at door ways, windows, and the like. These elements are usually aligned with gravity. Pipe lines and large cylindrical structures such as storage tanks tend to have their axes either horizontally or vertically aligned.

Vertical and horizontal surfaces and edges occur with such frequency in cultural objects that it is valuable to work out special case constraints for these construction elements. We intend to address the important special cases of plane and cylindrical structural elements, especially right parallelipipeds and right circular cylinders aligned with gravity, vertical and horizontal surfaces. To ignore these capabilities for use of special structures is throw away valuable information. Liebes is working to quantify these special structure constraints.

Consider the case of orthogonal trihedral vertices (OTVs), which appear as internal and external corners of right parallelipipeds. The approach is based upon the use of projective invariants. Images of OTVs show projective distortion depending upon their orientation and range relative to the camera. The study of projective transformations and stereoscopic imagery has yielded valuable formulations involving projective invariants, coordinate representations, and stereo edge element organization and analysis. These formulations have been applied to projective invariants of OTVs, using the locations of the vanishing points for their edges, or equivalently their surface normals. In an application to the important special case of nadir-oriented stereo cameras, a simple set of projection and visibility rules have been found that uniquely label the corners. Given an OTV in one image, the rules specify the quantitative appearance of the corresponding OTV in the conjugate image, as a function of relative displacement along the associated epipolar line. The nadir viewing case extends directly to oblique viewing configurations.

The simplicity of the rule formulation in nadir viewing arises from the facts that the vertical edge vanishing point coincides with the nadir point, and both of the remaining OTV edge vanishing points are oriented at right angles to one another at infinite distance parallel to the film plane. In the more general oblique case, all three vanishing points are at finite distance from one another in the film plane. The rules in the latter circumstance more explicitly utilize the projective relationship of the edges of the sixteen different kinds of corners to the vanishing points. Liebes has demonstrated that elements of the approach extend to the case of vertical cylinders with arbitrary polygonal cross section.

References

[Arnold 80] Arnold, R.D., Binford, T.O.; *"Geometric Constraints in Stereo Vision"* Proc SPIE, San Diego, Cal, July 1980.

[Binford 81] T.O.Binford; *"Inferring Surfaces from Images"*; Artificial Intelligence Journal forthcoming, 1981.

[Brooks 79] Brooks, R.A., Greiner, R., Binford, T.O.; **"ACRONYM: A Model-Based Vision System"**; *Proc Int Jt Conf on AI, Aug 1979.*

[Brooks 79b] Brooks, R.A.; "Goal-Directed Edge Linking and Ribbon Finding"; Proc Image Understanding Workshop, Palo Alto, Calif, Apr 1979.

[Brooks 80] Brooks, R.A., Binford, T.O.; *"Interpretive Vision and Restriction Graphs"*; Proc First National AAAI Conference, Stanford, Calif, August 1980.

[Brooks 81] R.A. Brooks; *"Model-Based Three-Dimensional Interpretations of Two-Dimensional Images"*; Proc Image Understanding Workshop, April 1981.

[Clarkson 81] K.L. Clarkson; *"A Procedure for Camera Calibration"*; Proc Image Understanding Workshop, April 1981.

[Clowes 71] Clowes,M.B.; *"On Seeing Things"*; AI Journal, 1971.

[Gennery 79b] Gennery, D.B.; *"Stereo Camera Solver"*; Proc Image Understanding Workshop Nov 1979, USC, Los Angeles.

[Henderson 79] R.L.Henderson, W.J.Miller, C.B.Grosch; *"Automatic Stereo Reconstruction of Man-Made Targets"*; SPIE Proc. Huntsville, Aug 1979.

[Horn 72] B.K.P.Horn; *"The Binford-Horn Edge Finder"*; MIT AI Memo 285, 1972, revised December 1973.

Huffman 71] Huffman,D.A.; *"Impossible objects as nonsense sentences"*; Machine Intelligence 6, 1971.

[Liebes 81] S.Liebes, Jr.; *"Geometric Constraints for Interpreting Images of Common Structural Elements: Orthogonal Trihedral Vertices"* Proc Image Understanding Workshop, April 1981.

[Lowe 80] Lowe, D.G.; *"Solving for the Parameters of Object Models from Image Descriptions"*; Proc Image Understanding Workshop, Univ of Md, April 1980.

[Lowe 81] D. Lowe, T.O.Binford; *"The Interpretation of Geometric Structure from Image Boundaries;"* Proc Image Understanding Workshop, April 1981.

[Mackworth 73] Mackworth, A.K.; *"Interpreting Pictures of Polyhedral Scenes"*; AI Journal 4, (1973).

[MacVicar-Whelan 81] P.MacVicar-Whelan, T.O.Binford; *"Line-finding to sub-Pixel Precision"*; Proc Image Understanding Workshop, April 1981.

[Marr 77] D. Marr, T. Poggio; *"A Theory of Human Stereo Vision,"* AI Memo 451, MIT, Nov 1977.

[Marr 79] D.Marr, E.Hildreth; *"Theory of Edge Detection"*; AI Memo 518, AI Lab MIT, April 1979.

[Miller 81] A. Miller, M. Lowry; *"General purpose VLSI chip with fault tolerant hardware for image processing"*; Proc Image Understanding Workshop, April 1981.

[Nelson 77] J I Nelson, H Kato, & P O Bishop; *"Discrimination of orientation and position disparities by binocularly activated neurons in cat striate cortex"*; J Neurophysiology, 40(2):260-283 1977.

[Nevatia 78] Nevatia, R., K.R.Babu; *"Linear Feature Extraction;"* Proc. ARPA Image Understanding Workshop, Pittsburgh, Nov.1978, 73-78.));

[Turner 74] Turner,K.; *"Computer Perception of Curved Objects using a Television Camera"*; PhD, Univ of Edinburgh, 1974.

[Ullman 79] S. Ullman; *"The interpretation of structure from motion"*; *MIT-AI Memo 476, 1979.*

[Waltz 72] Waltz,D.; "Generating Semantic Descriptions from Drawings of Scenes with Shadows"; MIT-AI Tech Rept AI-TR-271, 1972.also *"Understanding Line Drawings of Scenes with Shadows"*; P.H.Winston, ed; McGraw-Hill, 1975.

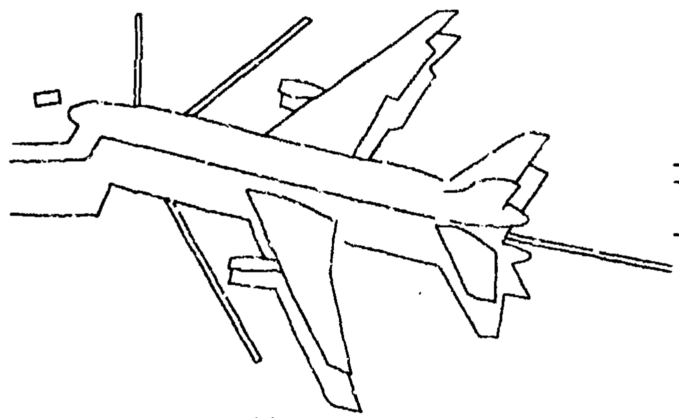


Figure 1a

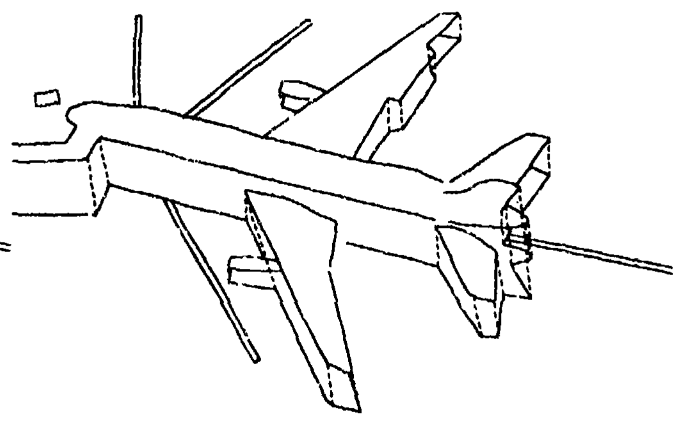


Figure 1c

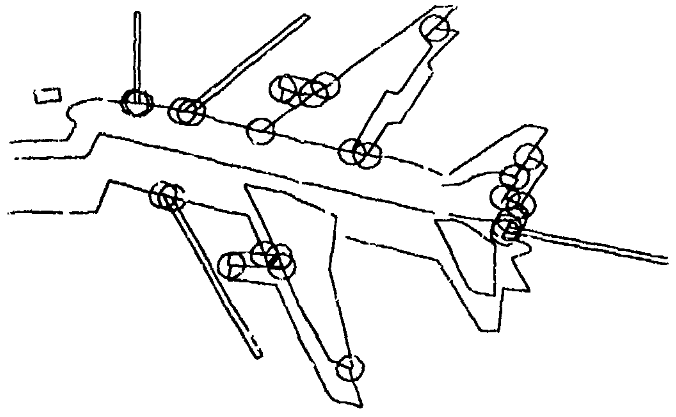


Figure 1b

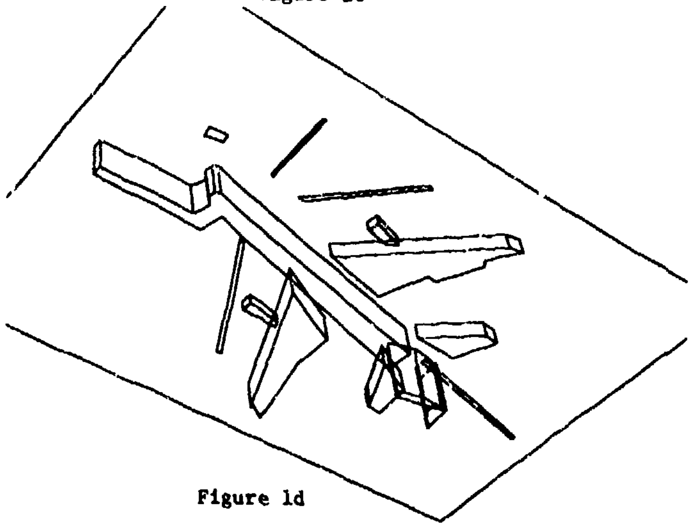


Figure 1d

PROGRESS IN THE USC IMAGE UNDERSTANDING PROGRAM

R. Nevatia and A.A. Sawchuk
Image Processing Institute
University of Southern California
Los Angeles, California 90007

Our goals have been to develop techniques that have wide utility and to test them on problems of image to map correspondence and scene matching. This paper is a summary of our program for the last year. Detailed of these projects may be found in [1-2].

SCENE MATCHING

We have continued to work on the problems of matching two images of a scene, or one image with a map, by generating symbolic descriptions from each. We have implemented matching techniques using search as well as relaxation techniques. New results using the latter are described in an accompanying paper [3]. We feel that our system can handle complex aerial images, the major limitations being due to failures of the segmentation procedures. We have initiated work to use the map as a guide to such segmentation.

SYMBOLIC TEXTURE ANALYSIS

We have a system in highly developed form for describing natural textures in terms of their primitives and relations among them. The descriptions consist of the primitive sizes and repetition patterns if any. A statistical analysis of our technique agrees with the experimental results. The descriptions have been tested for texture identification and we are investigating their use to estimate surface orientation form texture gradients. This technique is described in detail in another accompanying paper [4].

TEXTURE SYNTHESIS AND ANALYSIS

We have been working on several different statistical techniques for synthesizing natural textures. Both gray level and binary textures can be synthesized, and ten distinct techniques with various tradeoffs have been explored. The tradeoff parameters include such factors as computation time for generation, computation time for data collection, memory requirements, and quality of simulation. Many commonly occurring natural textures have been adequately simulated using very simple models, providing potentially great information compression for many applications. Other textures with macrostructure and nonstationary characteristics require more extensive computation to synthesize realistic, visually pleasing results. Although the success of any synthesis method is highly dependent on the texture itself and the modeling scheme chosen, general guidelines for predicting the performance of various techniques have been developed.

We also hope to use these techniques for texture classification and image segmentation. Some preliminary experiments employing statistical feature selection and classification techniques for discrimination have been undertaken by this approach. Another paper included in these proceedings [5] describes the texture synthesis results in more detail. Additional detail is contained in a forthcoming semi-annual technical report [2].

OTHER PROJECTS

We are continuing to make improvements to our road finding and linear feature extraction programs. We have incorporated Laplacian-Gaussian masks, suggested by Marr, as an alternative for low level edge extraction. We have also implemented techniques of connecting segments to give more continuous boundaries.

We are also investigating the use of hierarchical gradient relaxation techniques for matching of 2-D and 3-D shapes. We hope to present results in a later paper.

HARDWARE IMPLEMENTATION

In continuing work with Hughes Research Laboratories, Malibu, California, we are exploring architecture and hardware issues in the implementation of image understanding algorithms by VLSI techniques. The initial part of the study has concentrated on three algorithms: a) Nevatia-Babu Line Finder [6]; b) Ohlander-Price Region Segmentor [7]; and c) Laws Texture Analysis System [8]. These three algorithms are all very computation intensive and have a broad range of applications in image understanding research. Common to algorithms a) and c) are extensive two-dimensional convolutional processing, especially in the early stages of the algorithm. This convolutional processing is largely local and is well matched to the nature of VLSI systems, in which interconnections are difficult to implement.

More recently, the convolution problem has led to a detailed design for a 5 x 5 pixel convolution processor based on residue arithmetic.

The system is called RADIUS (Residue Arithmetic Digital Image Understanding System). The residue arithmetic approach has the advantages of modularity, ease of design, programmability, broad application to many problems, and ease of implementation in many integrated circuit technologies with submicron structures. Using residue arithmetic, there are no carries in the numerical computation and minimal interconnections on the chip are required. Very high speed processing is possible because many of the numerical operations reduce to table lookups in binary digital RAM's. Special purpose integrated circuits to perform part of the processing are being fabricated, and the construction of an experimental system is in progress. In addition to the increased computational speed for convolution, the processor has applications in evaluation of polynomial functions, integer coefficient transforms, enhancement operations, and moment calculations. Additional details are contained in a paper in these proceedings [9].

REFERENCES

1. R. Nevatia and A.A. Sawchuk, "Image Understanding Research, Semiannual Technical Report," USCIPI Report #990, Sept. 1980.

2. R. Nevatia and A.A. Sawchuk, "Image Understanding Research, Semiannual Technical Report," USCIPI Report #1010, March 1981.

3. K. Price, "Relaxation Matching Applied to Aerial Images," ARPA Image Understanding Workshop, Washington, D.C., April 1981 (these proceedings).

4. F. Vilnrotter, R. Nevatia and K. Price, "Symbolic Analysis of Natural Textures," ARPA Image Understanding Workshop, Washington, D.C., April 1981 (these proceedings).

5. D. Garber and A.A. Sawchuk, "Computational Models for Texture Synthesis and Analysis," ARPA Image Understanding Workshop, Washington, D.C., April 1981 (these proceedings).

6. R. Nevatia and K.R. Babu, "Linear Feature Extraction," Proceedings of the ARPA Image Understanding Workshop, Pittsburgh, Pa., November 1978, pp. 73-78.

7. R. Ohlander, K. Price and R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Computer Graphics and Image Processing, vol. 8, 1978, pp. 313-333.

8. K. Laws, "Textured Image Segmentation," USCIPI Report #940, January 1980.

9. S.O. Fouse and G.R. Nudd, G.M. Thorne-Booth, P.A. Nygaard and F.D. Gichard, "A Residue Based Image Processor for VLSI Implementation," ARPA Image Understanding Workshop, Washington, D.C., April 1981 (these proceedings).

PROGRESS AT THE ROCHESTER IMAGE UNDERSTANDING PROJECT

J. A. Feldman
K. R. Sloan, Jr.

The University of Rochester
Rochester, New York 14627

1. Technical Contributions

1.1. Parameter Networks and the Hough Transform

One of the most difficult problems in vision is segmentation. Recent work has shown how to calculate intrinsic images (e.g., optical flow, surface orientation, occluding contour, and disparity.) These images are distinctly easier to segment than the original intensity images.

The Hough transform idea has been developed into a general control technique. Intrinsic image points are mapped (many to one) into "parameter networks" [Ballard, 1980]. This theory explains segmentation in terms of highly parallel cooperative computation among intrinsic images and a set of parameter spaces at different levels of abstraction.

Our work with two-dimensional Hough transform techniques has been reported previously [Ballard, 1979; Sloan and Ballard, 1980]. The most recent application of these ideas is to the problem of detecting the presence and orientation of rigid, three-dimensional objects [Ballard, 1981; Ballard and Sabbah, 1981].

Hough-like techniques involving high-dimensional transform spaces have prompted a need for Dynamically Quantized Spaces. We have recently developed a data structure, based on the pyramid, which can cover a parameter space with a limited number of accumulators in such a way that fine precision is maintained, where it is needed. This data structure has the advantage that its resource allocation and connections are fixed. It differs from the usual pyramid in that the boundaries of elements in the pyramid are continually modified by a hierarchical warping process. Essentially, each cell tries to track the mean position of votes in its part of the space. This estimate of the local mean is used to define the boundaries of the cell's position [Sloan, 1981]

1.2 Computing with Connections

There is a rapidly growing interest in problem-scale parallelism, both as a model of animal brains and as a paradigm for VLSI. Work at Rochester has concentrated on connectionist models and their application to vision. The framework is built around computational modules, the simplest of which are termed p-units. We have developed their properties and shown how they can be applied to a variety of problems .[Feldman and Ballard, 1980].

To show how the framework can be applied to computational problems in vision, three specific examples have been developed in some detail. In the first, spatially distributed data can be associated with a complex concept. The second problem is related to eye movements, namely, how the eyes can be directed to move to interesting spatial features and how to avoid "reparsing" the images when those movements occur. Finally, we have considered the shape from shading problem and shown how a global parameter, such as light source position, interacts with the calculation of a spatially distributed parameter such as surface orientation.

1.3 Shape

A convenient representation for blob-like figures in an image consists of the orientation, length, and width of a bounding rectangle. One fast algorithm for producing such a bounding rectangle is based upon a dot product space. The analysis of the dot product space representation has been improved to handle certain pathological cases, and has been generalized to accommodate different criteria for the goodness of the representation [Sloan, 1980].

This simple concept of shape has been applied to the problem of reconstructing three-dimensional surfaces from very sparse data. The key idea is to use appropriate shape descriptors to hypothesize a transformation which accounts for the difference in shape between successive contours. When the hypothesized transformation is minor, very simple-minded surface reconstruction techniques are sufficient. When there are major differences in shape or position between successive contours, our method hallucinates new contours, using the hypothesized shape transformation [Sloan and Hrechanyk, 1981].

1.4. Adaptive Operators

Control is a crucial issue in Image Understanding. We have been investigating the role of low-level adaptive operators in both the analysis of aerial images and in problem solving. The aerial image work is reported on elsewhere in these proceedings [Selfridge and Sloan, 1981].

In general, problem solvers cannot hope to create plans that are able to fully specify all the details of operation beforehand and must depend on run-time modification of the plan to insure correct functioning. Fortunately, many primitive actions are highly stereotyped and can be performed by adapting pre-programmed tactics to the current goal context and operating environment. The architecure and operation of Adaptive Modules [Russell, 1981] demonstrates this approach to handling the problem of executing a low-level plan in a manner which uses multiple, parallel and distributed sources of knowledge.

1 5 Medical Applications

A system has been built in which Computer Tomograms of the human abdomen are searched as a 3-D image and matched against a detailed geometrical model of the abdomen anatomy. Detected organ boudaries serve to construct an instance of the model that reflects the actual anatomy of a particular patient as revealed by the corresponding image data.

The model-directed approach makes possible the detection of hard-to-find organs (e.g., kidneys) based on known locations of easy-to-find organs (e.g., spinal column), thus relaxing the problem of obscured boundaries in noisy data that tend to hinder data-directed approaches.

The model is hierarchical, built of Generalized Cylinders, and is inherently parallal. It captures relational, structural, and quantitative knowledge that is represented as both data and procedures [Shani, 1980].

2. System Support

2.1. Hardware

The Grinnell GMR-26 display device is DMA-interfaced to an Eclipse computer, and has been invaluable as an output device for our experiments. An Optronics Colorscan C-4100 drum scanner is on site and interfaces to the Vision Eclipse.

Both Eclipse computers are fully configured and have been running effectively with our distributed system software. A VAX 11/780 (purchased with non-DoD funds) is operating and has been integrated into the local network. A new, larger capacity Eclipse has been added to the gateway configuration, giving greater capacity and reliability. We are currently installing several additional personal computers and a laser printer.

3.2. Software

We have been working closely with other IU contractors (particularly CMU) to develop a uniform communication facility for use in the testbed.

Local image-processing and graphics software has been transferred to the VAX, in C.

The external representation for PLITS style messages has been specified, and is being implemented for transmission of self-describing messages. A Name-Type-Value (NTV) message is conceptually an unordered set of triples consisting of a name, a data-type, and a value of the given data-type. These triples are called slots. The external format of these messages is used by the communication subsystems. We make no other assumptions about other information which may be required by the communication subsystem (e.g., headers or checksums), nor do we require that a message fit inside a single physical packet. Logically we model the external format of a message as a variable length vector of octets (eight bit bytes) [Low, 1980].

REFERENCES

Ballard, D.H., Generalizing the Hough Transform to Detect Arbitrary Shapes, TR55, Computer Science Department, University of Rochester, October, 1979 (a); Pattern Recognition (to appear).

Ballard, D.H., Strip Trees: A Hierarchical Representation for Curves, CACM, (to appear).

Ballard, D.H., Parameter Networks: Towards a Theory of Low-Level Vision, TR75, Computer Science Department, University of Rochester, November 1980.

Ballard, D.H., and D. Sabbah, Detecting Object Orientation from Surface Normals, submitted to IJCAI.

Feldman, J.A., A distributed information processing model of visual memory, TR52, Computer Science Department, University of Rochester, December, 1979.

Low, J.R., Name-Type-Value Protocol, TR73, Computer Science Department, University of Rochester, July 1980.

Russell, D.M., Adaptive Modules for Low Level Problem Solving, (in preparation).

Selfridge, Peter G. and K.R. Sloan, Jr., Locating Objects Under Different Conditions: An Example in Aerial Image Understanding, (to appear) Proceedings: Pattern Recognition and Image Processing, Dallas, Texas, August 1981.

Selfridge, Peter G. and K.R. Sloan, Jr.,Reasoning About Images: Application to Aerial Image Understanding, Image Understanding Workshop, April 1981.

Shani, U., A 3-D Model-Driven System for the Recognition of Abdominal Anatomy from CT Scans, TR77,Computer Science Department, University of Rochester, May 1980.

Sloan, K.R.,Jr., Analysis of 'Dot Product Space Shape Descriptions, TR74, Computer Science Department, University of Rochester, June 1980.

Sloan, K.R.,Jr. and L.M. Hrechanyk, Surface Reconstruction from Sparse Data, (to appear) Proceedings: Pattern Recognition and Image Processing, Dallas, Texas, August 1981.

Sloan, K.R., Jr., Dynamically Quantized Pyramids, (in preparation).